Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | Thesis and Dissertation Collection

1976

Study and design of flight data recording systems for military aircraft.

Baetz, Lloyd Norman

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/17955

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

STUDY AND DESIGN OF FLIGHT DATA RECORDING SYSTEM FOR MILITARY AIRCRAFT

Lloyd Norman Baetz

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

STUDY AND DESIGN OF
FLIGHT DATA RECORDING SYSTEMS
FOR MILITARY AIRCRAFT

by

Lloyd Norman Baetz

June 1976

Thesis Advisor: U. R. Kodres

Approved for public release; distribution unlimited.

T174010

SECURITY CLASSIFICATION OF THIS PAGE (When Date Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) Study and Design of Flight Data Recording Systems for Military Aircraft	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis June 1976
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s) Lloyd Norman Baetz	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE June 1976
	13. NUMBER OF PAGES 128
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Flight data recording
nonvolatile solid state memory
microprocessors
data compression
inertial navigator

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Investigation of aircraft wreckage does not provide crash investigators with adequate information. Crash-protected flight recorder data is invaluable when determining accident cause factors. Inertial navigation systems provide an excellent source of highly accurate flight parameters. Nonvolatile solid state memory is available which can replace failure prone magnetic tape recording in flight

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

SECURITY CLASSIFICATION OF THIS PAGE (When Date Entered)

recorder systems. Microprocessors are available with the capability of compressing flight data for solid state memory storage. Data compression trials indicate that a flight data recording system using microcomputer preprocessing and nonvolatile solid state memory is feasible.

Study and Design of Flight Data Recording Systems for Military Aircraft

by

Lloyd Norman Baetz
Captain, Canadian Armed Forces
B.E.Sc., University of Western Ontario, 1970

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1976

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940

ABSTRACT

Investigation of aircraft wreckage does not provide crash investigators with adequate information. Crash-protected flight recorder data is invaluable when determining accident cause factors. Inertial navigation systems provide an excellent source of highly accurate flight parameters. Nonvolatile solid state memory is available which can replace failure prone magnetic tape recording in flight recorder systems. Microprocessors are available with the capability of compressing flight data for solid state memory storage. Data compression trials indicate that a flight data recording system using microcomputer preprocessing and nonvolatile solid state memory is feasible.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to all those who provided background material and shared their experience in the flight recorder and crash investigation fields. In particular Rodney Wingrove of NASA Ames Research Center, Dave Althaus of Lockheed Aircraft Service Company, Robert Foley of Hamilton Standard, and especially Carol Roberts of the National Transportation Safety Board. I am deeply grateful to my thesis advisors Dr. Uno Kodres and Capt. Clyde Tuomela for their patience, guidance and timely advice throughout this work. Finally, a very special thank you to my wife, Myrna, for her patience and understanding.

TABLE OF CONTENTS

I. INTRODUCTION - - - - - - - - - - - - - - - - - - - 9

II. AIRCRAFT FLIGHT DATA RECORDERS - - - - - - - - - - 11

A. HISTORY OF FLIGHT DATA RECORDING - - - - - - - 11

B. FLIGHT DATA RECORDING EQUIPMENT- - - - - - - - 14

C. DATA RECOVERY AND ANALYSIS - - - - - - - - - - 18

III. FUTURE RECORDING SYSTEMS - - - - - - - - - - - - 23

A. INDUSTRY PROPOSALS - - - - - - - - - - - - - - 23

B. A PROPOSED SYSTEM FOR MILITARY AIRCRAFT- - - 26

1. Design Restrictions- - - - - - - - - - - - 26

2. Solid State Memory - - - - - - - - - - - - 27

3. Microcomputer Preprocessing- - - - - - - - 30

IV. DATA COMPRESSION TRIALS- - - - - - - - - - - - - 34

V. CONCLUSIONS AND RECOMMENDATIONS- - - - - - - - - 47

APPENDIX A: SOLID STATE MEMORY TECHNOLOGY - - - - - - - 48

APPENDIX B: PROGRAM TO LIST ORIGINAL DATA - - - - - - - 59

APPENDIX C: PROGRAM TO COMPRESS DATA- - - - - - - - - - 68

APPENDIX D: PROGRAM TO RECOVER AND LIST COMPRESSED DATA - - - - - - - - - - - - - - - 90

APPENDIX E: PROGRAM TO PLOT ORIGINAL AND COMPRESSED DATA - - - - - - - - - - - - - -101

APPENDIX F: FLIGHT PARAMETER LISTINGS AND PLOTS - - -106

BIBLIOGRAPHY -125

INITIAL DISTRIBUTION LIST- - - - - - - - - - - - - - - -128

LIST OF ACRONYMS

AIDS	Aircraft Integrated Data Systems
ALU	Arithmetic Logic Unit
ARINC	Aeronautical Radio, Inc.
ATC	Air Transport Control
BEAMOS	Beam-Addressable Metal Oxide Semiconductor
BORAM	Block-Addressable Random Access Memory
CADC	Central Air Data Computer
CCD	Charge-Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DEC	Digital Equipment Corporation
DFDR	Digital Flight Data Recorder
EAROM	Electrically-Alterable Read Only Memory
ECL	Emitter-Coupled Logic
FAA	Federal Aviation Administration
FAMOS	Floating-Avalanche Metal Oxide Semiconductor
FDAU	Flight Data Acquisition Unit
FDEP	Flight Data Entry Panel
FDR	Flight Data Recorder
GMT	Greenwich Mean Time
I^2L	Integrated-Injection Logic
INS	Inertial Navigation System
LARAM	Line-Addressable Random Access Memory

LSI	Large Scale Integration
MNOS	Metal Nitride Oxide Semiconductor
MOS	Metal Oxide Semiconductor
MOSFET	Metal-Oxide-Silicon Field-Effect-Transistor
NMOS	N-Channel Metal Oxide Semiconductor
NTSB	National Transportation Safety Board
PMOS	P-Channel Metal Oxide Semiconductor
RAM	Random Access Memory
ROM	Read Only Memory
R/W RAM	Read/Write Random Access Memory
SOS/MOS	Silicon-On-Sapphire Metal Oxide Semiconductor
TTL	Transistor-Transistor Logic

I. INTRODUCTION

The complexity and cost of military aircraft have continued to increase along with demands for higher performance and safety. Unfortunately accidents continue to occur and in some instances two, three and even more aircraft of the same type have crashed before the cause can be determined. This loss of high value aircraft and the possibility of crew injury or death has been allowed to continue while a proven source of accident information continues to be overlooked.

The use of recorded flight data in accident investigation has expanded widely in recent years and in many instances has been the only source of evidence which could be used to establish the cause of a crash. Crash-protected flight data recorders have been required on large civilian aircraft since 1957 but are still not carried on most military aircraft.

This thesis is a study of the development of flight data recording systems and their design for military aircraft. Section II is a history of flight data recording including a description of equipment and data recovery methods. Section III describes future flight data recording systems and a proposed design for use in military aircraft. Section IV describes the data compression trials which were carried out. The conclusions and recommendations of this report are presented in Section VI. Appendices A to F include a description of solid state memory technology and the programs used

in the data compression trials along with listings and plots of original and compressed data.

II. AIRCRAFT FLIGHT DATA RECORDERS

A. HISTORY OF FLIGHT DATA RECORDING

Since the early days of aviation, flight data recording has progressed from simple handwritten notes to highly sophisticated digital systems which record millions of measurements during the period of a single flight. The recorders principal use, until recent years, was for flight test monitoring and the acquisition of airworthiness data. It is now used extensively in accident investigations. The role of flight recording in aircraft accident investigation and prevention is summarized in Refs. 1, 2 and 3.

For many years the precrash condition and performance of an aircraft, which had been involved in an accident, was derived from examination of the wreckage, studying maintenance records, weather information, flight operation data and human factors. Ground witnesses were the primary source of information concerning the aircraft's flight path and maneuvers. The difficulties of wreckage analysis vastly increased with the introduction of high performance aircraft. The higher operating speeds and greater structural mass greatly increased the release of energy and the extent of disintegration at the time of impact. In addition, higher altitude flight and longer flight times reduced post-crash knowledge of the operational features of the flight.

As a result of widely expressed demands by investigating authorities and flight safety organizations, regulations were established for the carriage of flight data recorders on large civilian aircraft. These mandatory requirements gave considerable momentum to the further development of advanced flight data recording systems and their use in accident investigation and performance monitoring.

The following is a brief history of the development of Federal Aviation Administration (FAA) regulations governing flight recorder utilization and technical standards [Refs. 4 and 5].

The first civil air regulation on flight recorders was issued in April 1941. It required air-carrier aircraft to record altitude and whenever the radio transmitter was turned on or off. The compliance date was delayed a number of times until finally in June 1944 the requirement was rescinded due to maintenance difficulties and lack of replacement parts for the recorders. A similar regulation was issued in September 1947 requiring aircraft of 10,000 pounds or more to record altitude and vertical acceleration. Again this regulation was rescinded in July 1948 due to lack of suitable recording equipment.

Finally in 1957, after nine years of study, the Civil Aeronautics Board adopted regulations requiring flight recorders to be installed, by September 1957, in all air-carrier aircraft which were over 12,000 pounds and operated at altitudes above 25,000 feet. The parameters to be recorded

were airspeed, altitude, direction, vertical acceleration and time. In September 1959, these regulations were amended to require the retention of flight records for 60 days and the operation of the flight recorder from the beginning of the takeoff roll to the end of the landing roll.

The regulations were amended again, effective September 1972, to require the recording of data from which the time of each radio transmission to air traffic control could be determined. Effective September 1973, all large aircraft which were certified after September 1969 and operate above 25,000 feet or are turbine powered were required to be equipped with an expanded parameter flight recorder. The additional parameters required were: pitch attitude, roll attitude, sideslip angle or lateral acceleration, pitch trim position, control column or pitch control surface position, control wheel or lateral control surface position, rudder pedal or yaw control surface position, thrust of each engine, position of each thrust reverser, and trailing edge flap or cockpit flap control position. Effective March 1974, each recorder had to be equipped with a device to assist in locating the recorder under water.

These regulations have resulted in the installation of crash protected flight recorders in all large civilian transport aircraft registered in the United States. The flight data recorder has added a new dimension to the investigation of accidents by supplying detailed information about pre-crash conditions. It has increased the speed and accuracy of

accident investigations and has made possible analysis of the complex interactions between the flightcrew, the aircraft and the environment.

B. FLIGHT DATA RECORDING EQUIPMENT

At present there are two types of crash protected flight data recorders used by United States civilian air-carriers. The older type flight data recorder (FDR) has electromechanically operated styli which scribe a permanent record of the data on a metal foil recording medium. Pressure altitude, indicated airspeed, magnetic heading and vertical acceleration are recorded against a base of elapsed time. The newer type digital flight data recorder (DFDR) records a much wider range of aircraft flight data on magnetic tape. A flight data acquisition unit (FDAU) is used to access analog data from various sensors and transmitters in the aircraft and convert the data to digital form for transmission to the DFDR.

A typical FDR is the Lockheed Aircraft Service Company MOdel 109-C [Ref. 6]. It is housed in a 15 inch diameter, insulated, stainless steel sphere and has an aluminum foil recording medium, which can record up to 200 hours of data. The foil is contained in a stainless steel cassette and is fed over a teflon coated platen, which is part of the cassette wall. The recording styli are mounted so that they contact the foil surface. Altitude and airspeed are sensed from the aircraft pitot and static pressure systems and the recording styli are positioned mechanically in response to

changes in these pressures. An alternate source of altitude and airspeed is the central air data computer (CADC). Magnetic heading is obtained from the #2 aircraft compass system and vertical acceleration is obtained from an accelerometer located close to the aircraft's center of gravity. Like the altitude and airspeed from the CADC, magnetic heading and vertical acceleration are servo signals which position the recording styli using servo motors. The Sundstrand Data Control Model F-542 and Fairchild Industrial Product Model 5424 are two other versions of the FDR.

A typical DFDR is the Lockheed Aircraft Service Company Model 209 [Ref. 7]. It records 1,670 bits of digital data per inch, at 0.46 inches per second, on mylar magnetic recording tape. There are six data tracks with over four hours of data on each track. Tracks 1, 3 and 5 are recorded in the forward direction and tracks 2, 4 and 6 in the reverse direction. After all six tracks have been used, and more than 25 hours of data have been stored, recording is resumed on track 1, erasing the previous data. This recorder is used with a FDAU which generates the timing signals required to define bit, word, subframe and frame times. Each frame of data contains four subframes, and each subframe contains 64 12-bit words representing one second of digital data. The first word of each subframe is a synchronization word, provided by the FDAU, which signals the start of a new subframe. The FDAU also converts the data to Harvard Bi-phase format, and transmits it to the DFDR in serial form. The Sundstrand

Data Control Model 573A is another version of the DFDR.

There are 3 companies in the United States who supply digital flight data systems using the Lockheed or Sundstrand DFDR. They are Garrett AiResearch, Hamilton Standard and Teledyne Controls. The recording system consists of a DFDR, FDAU, flight data entry panel (FDEP) and the required transducers and sensors throughout the aircraft. The FDEP allows the flightcrew to enter documentary data such as flight number and date on the recording medium. These systems are being used primarily on the new generation of wide-bodied aircraft, the Boeing B-747, Douglas DC-10 and Lockheed L-1011.

The main concern in recorder design, aside from the operational and recording accuracy requirements, is survivability of the recording medium in an accident. Protection must be provided against the crushing, penetration and acceleration forces of impact or explosion. The recorder must also be able to survive exposure to fire, immersion in sea water and the chemical attack of hydraulic, de-icing and fire extinguishing fluids, fuels and acids. Survival of the recording medium is primarily ensured by built-in protection, but the recorders location in the aircraft was also found to be an important factor. Experience during the early years of accident investigation indicated that the rear fuselage and tail structure are most likely to survive, or be least severely damaged, even in a major accident. Federal Aviation Regulations requiring recorders to be moved as far aft in the fuselage as possible have greatly improved the recording mediums chances of survival.

The flight recorders used in United States military aircraft are installed in an ejectable airfoil package to help ensure their survivability. The airfoil system, developed by Leigh Instrument Limited, is being used in the USAF C-135, C-141 and C-5A transport aircraft and is being installed in the USN P-3C.

When crash sensors, which are located in the aircraft's wingtips, nose and undercarriage area, detect aircraft structural breakup or deformation, they cause the airfoil package to eject from the aircraft. The package is mounted as far aft as possible to ensure maximum probability of survival. If the airfoil does not eject before the aircraft strikes the ground the rear structure will normally retain its forward velocity long enough to ensure a good airfoil departure. Airflow over the airfoil then allows it to generate lift and fly away from the crash site. The aerodynamic properties of the airfoil cause it to fly in an arc and rapidly slow to terminal velocity and achieve a safe landing. The airfoil contains a radio beacon which is activated when the airfoil is separated from the aircraft. It transmits an emergency distress signal to aid location of the crash site and the flight recorder.

The Leigh Instrument Limited AN/ASH-20(V) flight recorder/locator system is being installed in the P-3C aircraft [Ref. 8]. This system includes a magnetic tape recorder, a beacon locator, a recorder electronics unit and a recorder control unit. The tape recorder and beacon locator are

contained in the airfoil package described above. The 8-track, bi-directional, cassette tape recorder preserves 30 minutes of flight data and audio signals. Four tracks (three audio and one data) are utilized in each direction. The recorder electronics unit samples 32 channels of data from the aircraft instruments and special transducers and processes it into 9-bit digital words. It also amplifies audio signals from the aircraft interphone system before they are recorded. The recorder control unit is located in the flight deck and allows crew members to monitor system status and voice recording quality. The beacon locator system will transmit a 250 milliwatt, omnidirectional signal for a minimum of 48 hours and can be detected at a range of more than 50 miles from a height of 10,000 feet.

The system is being installed to aid accident investigation personnel in determining crash causes, to help prevent similar crashes, to aid search aircraft in locating and rescuing aircrew from crashed aircraft and to provide aircraft maintenance information.

C. DATA RECOVERY AND ANALYSIS

The equipment required to recover data from the recording medium depends upon the type of flight recorder used and ranges from high resolution coordinate measuring equipment to digital computers.

The data recorded on the metal foil of an FDR is recovered using a high accuracy optical readout machine. Measurements are made by following the scribed traces with a moving microscope and logging the X and Y coordinates in inches of

microscope movement. These values are then plotted on a graph with appropriate scales for analysis of aircraft operation.

Data from the DFDR is recovered using a computer based ground processing station. Since the DFDR tape is not compatible with the computer,the data must first be transcribed, in the correct format, onto a 9-track computer tape. This data, along with information on the airline and type of aircraft involved, may then be processed on the ground station computer of any large computer facility. Various analysis programs have been developed to reduce the data and display it in readable formats. These include second-by-second listings and plots of selected parameters versus time.

The National Transportation Safety Board (NTSB) has the responsibility of investigating civil aircraft accidents, reporting their probable cause and making safety recommendations to help prevent future accidents. During the 14 year period from 1959 to 1973 NTSB reviewed 509 accidents involving FDRs and four involving DFDRs [Ref. 6]. Recorder malfunctions or accident damage prevented readout of data in 8% of the FDR cases,but after they were relocated to the aft of the aircraft only one recorder received damage which prevented data readout.

NTSB has recently installed a complete data reduction station to process data from flight data recorders [Refs. 9 and 10]. The heart of the system is a minicomputer (PDP-11/40) with 24K of core memory and a disk operating system. Peripherals include a CRT terminal, two 9-track magnetic tape drives, a

high speed printer/plotter and a paper tape reader and punch. Specialized hardware includes two DFDR readers which reformat the Harvard bi-phase serial data into 9-track computer compatible format. There is also an interface to transfer the X-Y coordinate data from the FDR readout machine into the computer. The system software includes a program for converting raw data into the original parameter values and a search routine for locating a specific flight record. There are also limit exceedance, max-min, plotter and print routines. Interaction between the operator and computer is via the terminal in question-answer mode.

The NTSB plans to adapt an existing routine to the PDP 11/40 which will prepare a ground track of the aircraft from the recorded data. The flight recorder data is first corrected for estimated meteorological conditions,and any available radar or other position data,to give estimates of the geographical position of the aircraft, its heading, and ground speed. This will be very useful in cases involving thunderstorm activity, wake turbulence and midair collisions. It will also aid in determining whether the flightpath of the aircraft was consistent with its aerodynamic characteristics.

In several recent accidents involving large transport aircraft the investigators were highly dependent on data from the DFDR since examination of the wreckage gave no clue to the cause. In these cases DFDR data was sufficient to establish the accident cause factors. In one accident it was possible to use the recorded data to reconstruct the aircrafts

motion in space by employing the airframe manufacturers 6-degree-of-freedom computer simulation of the aircraft. The results showed that the flight path was consistent with the established aerodynamic characteristics of the aircraft. This led the investigators to conclude that the aircraft and its systems were not factors contributing to the accident.

The NASA Ames Research Center has recently investigated advanced data processing methods for combining FDR and DFDR data with Air Transport Control (ATC) radar recordings, wind and temperature profiles, aircraft aerodynamic data, etc. The basic objective was to develop the capability to derive a number of additional parameters which were not originally recorded. The preliminary results indicate that the derived quantities are in good agreement with the actual values [Ref. 11].

Ames Research Center has also used its flight simulator facilities to derive the wind forces which were acting on an aircraft. The DFDR, aerodynamic and engine data combined with the equations of motion of the aircraft were fed into the simulator. The simulators response was compared with the actual response of the aircraft and the forces required to make the two responses the same were attributed to wind. The advanced methods of data processing, developed at Ames Research Center, are applied in support of NTSB accident investigations.

The information recorded by flight recorders has become vitally important in the understanding of the subtle causal factors of aircraft accidents and the prevention of future

accidents. Wreckage no longer produces sufficient information to assess the causal factors of accidents. Data cannot be obtained by examining the complex hardware and avionics circuits, such as automatic flight control systems and navigation receivers, once power has been removed. Information retrieved from the flight data recorder has made more precise accident cause determination possible. It has also provided technical substantiation for recommended solutions to safety problems.

III. FUTURE RECORDING SYSTEMS

A. INDUSTRY PROPOSALS

Advances in flight data recorder technology, particularly those designed to Aeronautical Radio, Inc. (ARINC) standards, have made the recording of additional data technically and economically feasible. The increased recording capacity and wider parameter coverage of these recorders will undoubtedly extend their range of application.

The NTSB believes that additional information is essential to the conduct of thorough and expeditious accident investigations and has recommended the mandatory recording of additional parameters. Investigations of recent accidents involving wide-bodied transport aircraft have shown that additional parameters would have provided a more complete understanding of the underlying causal factors, and would have produced more effective measures to prevent future accidents.

The new generation of wide-bodied aircraft, which are being used by all major airlines, cannot be effectively maintained using standard maintenance procedures. In order to cope with the extreme complexity of these aircraft, the airlines have started using on-board and ground processing of flight data. Their main objectives are to reduce maintenance costs, improve aircraft availability, increase flight safety and increase the effectiveness of flight crews in aircraft operations.

To meet these data processing objectives,a wide range of aircraft integrated data systems (AIDS) have been developed for use by the airlines. They are built to meet ARINC characteristic 573, which specifies a standard installation for the basic flight recorder system and provides the expansion capability needed for a wide variety of AIDS [Ref. 12].

The primary role of AIDS is to record in-flight aircraft data for subsequent processing and analysis on the ground. Various system configurations may be employed to meet an airline's needs based on parameters of interest and the end use of the data. Applications range from crash-protected recording of flight data for accident investigation to on-board processing and recording of real-time data and further analysis at a ground processing center. Airlines may add just a few additional sensors or go to more complex AIDS by adding additional signal acquisition units, other recorders, a computer, data compression system, etc. [Ref. 13].

The airlines and NASA Ames Research Center have been successful in utilizing flight recorder data, voice recorder data, radar track and meteorological data to recreate aircraft accident conditions on flight simulators. However, in many cases the data available requires considerable extrapolation before it is suitable for use in the flight simulator. In order for flight simulators to become a truly effective accident investigation tool, data from advanced flight data recorders must be readily available. Conclusions and recommendations concerning the role of the flight simulator in aviation safety are contained in Ref. 14.

Most current high value aircraft are equipped with inertial navigators which are another potential source of data that could easily be adapted to flight simulators. If inertial navigator parameters were recorded on every flight, they could be used to improve the equations of motion used in simulator mathematical models. The simulator would then more closely represent real world characteristics and could more easily be used to recreate accident conditions. They would greatly increase the effectiveness of accident investigations by providing a means of testing various hypothesis of what occurred and should be used to validate solutions and corrective actions.

The U. S. Navy Test Pilot School has carried out a feasibility study to determine if an inertial navigation system (INS) could be utilized to derive the parameters which are conventionally used to describe an aircrafts motion [Ref. 15]. They found that accurate earth-referenced aircraft attitude information could be obtained directly from the INS. By transforming the North, East and vertical velocities from the INS into velocity components along the aircrafts three body axes, attitude rate, accelerations, velocities, angle of attack and sideslip were calculated. The North and East components of wind were determined by comparing air data computer true airspeed with INS airspeed. Results indicated that traditional parameters were easily obtained with higher accuracy and much less noise. Parameters normally not available could be computed and the cost of purchasing and installing sensors was eliminated.

Recently Hamilton Standard proposed a crash data retrieval system for military aircraft [Ref. 16]. They plan to use microcomputer preprocessing along with solid state mass data storage and large scale integrated and hybrid electronic circuits to reduce the cost, size and weight of their system. Lockheed Aircraft Service Company and Leigh Instruments Limited are also looking at solid state memory and microprocessors for possible use in their systems.

The importance of recorded flight data in crash investigation has been demonstrated again and again. The equipment and methods for highly complex flight recording systems are available. Unfortunately most military aircraft still do not carry flight recorders of any kind.

B. A PROPOSED SYSTEM FOR MILITARY AIRCRAFT

1. Design Restrictions

Most flight data recorders have been designed for use in large transport aircraft and so are not entirely suitable for smaller military aircraft. Their lower inertia and more rapid responses mean that extremes of maneuverability and attitude will often be encountered and the sampling rates of many parameters will have to be very high. There will also be size and weight restrictions on small military aircraft which were not as critical in transport aircraft. The precise position of the recorder in relation to surrounding items of high mass is also important, particularly in the rear-engine case. Finally the question of cost must be considered. In these days of multi-million dollar military

aircraft the relative cost of a crash-protected flight data recorder system will be very small, but it goes without saying that installation and maintenance costs must be kept as low as possible.

The proposed flight data recording system [Fig. 1] incorporates recent advances in computer and solid state memory technology to cope with the cost, size, and weight restrictions of military aircraft. It is also designed to cope with their extreme operating conditions as well as provide greater reliability and survivability of the recording medium.

2. Solid State Memory

The proposed system has a solid state recording medium rather than the standard magnetic tape. The tape recorder was eliminated since it is the most failure prone component in flight data recording systems. This poor reliability is due mainly to the fact that tape recorders are complex electromechanical devices with many moving parts. As a result they require periodic maintenance such as lubrication, head cleaning and alignment to prevent loss of data. The capability of tape recorders is also limited because of their fixed recording and playback rates and relatively long start and stop times. There are also the basic design problems of wow, flutter and jitter as well as skew in multitrack systems which contribute to the bit error rate.

High density nonvolatile solid state memories, capable of replacing tape recorders, are just now becoming available. They have the advantage of high reliability due

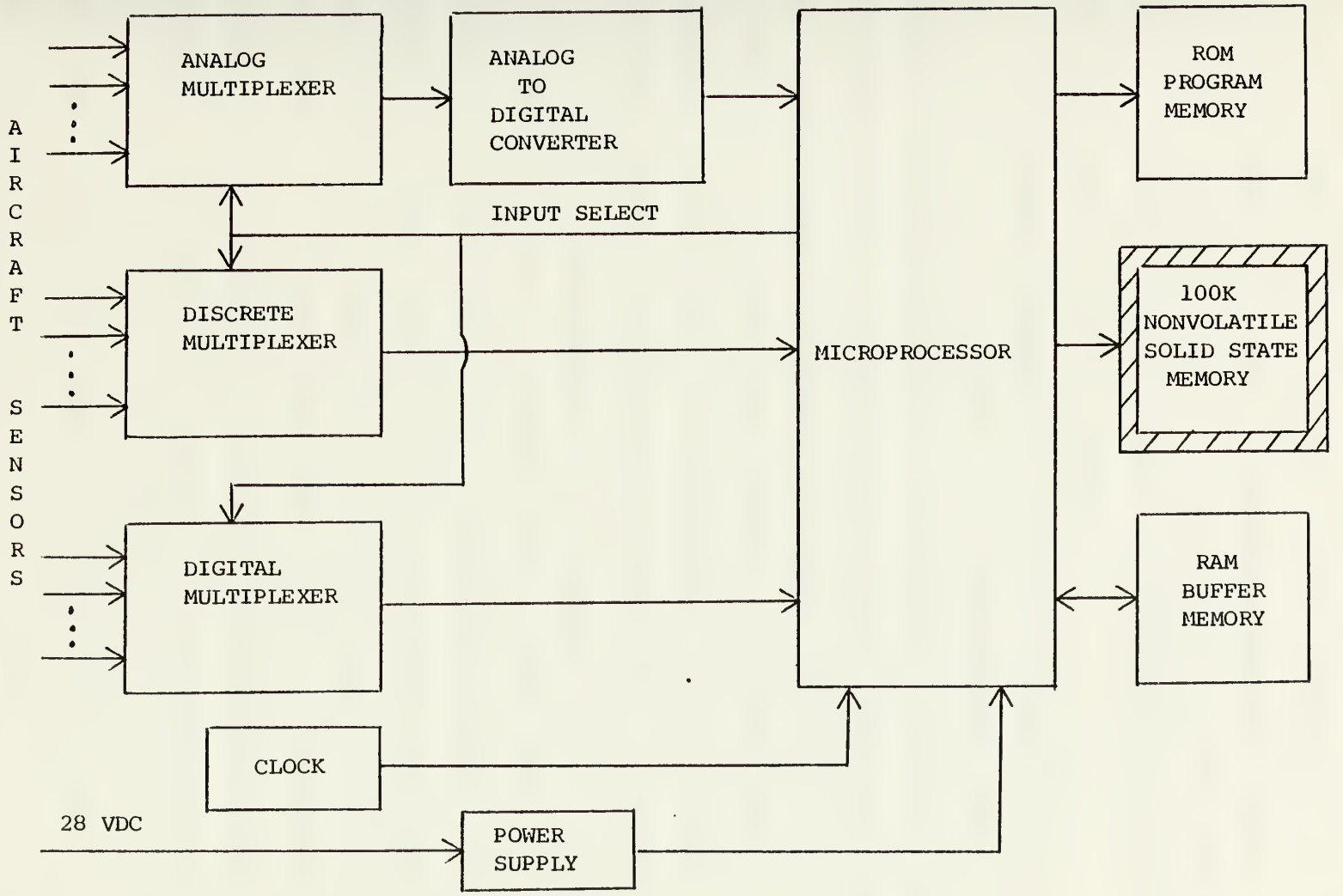


FIGURE 1 PROPOSED FLIGHT DATA RECORDER SYSTEM

to no moving parts and no maintenance requirements, as well as being easier to protect against crash damage. Solid state memory can easily handle bursts of data which may occur prior to a crash since its data rate is only limited by memory cycle time. Since the rate of data storage can be easily adjusted, only significant data need be stored and ground processing time is greatly reduced.

The memory technologies that best lend themselves to this role are: metal nitride oxide semiconductor (MNOS), floating avalanche metal oxide semiconductor (FAMOS), magnetic domain bubble, charge-coupled device (CCD) and complementary metal oxide semiconductor (CMOS). Each type has drawbacks and limitations but research and development is continuing in most areas. A description of the various solid state memory technologies is contained in Appendix A.

CCD and CMOS memories are basically volatile and so require battery power to prevent loss of data [Ref. 17]. They have the advantage of being relatively mature technologies with proven reliability and are readily available. FAMOS and MNOS are basically nonvolatile but only MNOS is being actively developed [Ref. 18]. Unlike conventional random access memories, MNOS must be erased before data is rewritten. Presently erase time is one to two seconds and the number of erase cycles is limited to from 10^6 to 18^8. Write time is also high (from one to two milliseconds) and chip density is low (1K to 2K bits). The goal of current research is a 1 microsecond write, a 1 microsecond erase and 10^{10} erase/write cycles before failure. Bubble memory

is by far the most promising technology to replace magnetic tape [Refs. 19 and 20]. Like MNOS it is nonvolatile and like CCD and CMOS it has very high storage density and does not require an erase cycle. Bubble memories are not presently available, but Hatachi, Ltd. of Japan has announced a 256K bit bubble memory, using 16K chips, which is to be available in 1976.

3. Microcomputer Preprocessing

Another new feature of the proposed system is microcomputer preprocessing of data. Tape systems are limited to a constant data rate because of the fixed recording speed and relatively long start and stop times. This inefficient use of memory cannot be tolerated with the relatively small solid state memory.

The microcomputer compresses the data by eliminating the recording of redundant data and data which changes within an acceptable tolerance [Section IV]. Since microprocessor instruction time is less than 10 microseconds and the number of parameters will be less than 100, a minimum of 1000 instructions will be available to process each parameter during a second. This should allow each parameter to be sampled up to 25 times per second,since only the basic arithmetic and logical functions are required. The only division and multiplication operations in the compression algorithm involve the number 64 so they can be carried out using right or left shifts.

The high sampling rate of this system will easily handle the rapid parameter changes which may occur prior to

an incident or crash. Since memory size is limited, the system must be set to provide the best balance of data rate to data retention time.

There are a number of excellent microprocessors which could be dedicated to this role. The lower cost of the one-chip NMOS microprocessor and its adequate capability indicates that it is the best choice. The bit-slice bipolar TTL microprocessor could also be used in a 12-bit word format to match the normal word size of the analog to digital converters.

A microprocessor results when an arithmetic logic unit (ALU) and central processing unit (CPU) control function are implemented on one, or a small number of large scale integrated (LSI) chips. When all five of the major computer subsystems; CPU·control, ALU, memory, input and output are contained in a small set of LSI packages, and a source of external power and timing clock are applied, the system is called microcomputer. The microprocessor contains the basic control logic, the circuitry for decoding instructions and the logic circuits for processing arithmetic functions. Both read/write random access memory (R/W RAM) and read only memory (ROM) types are required [Appendix A]. Most microprocessor programs are permanently encoded in ROM and data is stored in R/W RAM. Input/output circuity is used to interface the microcomputer with peripheral equipment.

The characteristics of interest in a microprocessor are controlled by the performance achieved using a particular fabrication method [Appendix A and Refs. 21-23]. The first

microprocessors were a byproduct of P-channel metal oxide semiconductor (PMOS) solid state memory research and were implemented on a standard 16 pin circuit package. To conserve space and pins, both internal and external interconnections were made using a 4-bit bidirectional multiplexed data bus. Since both address and data could not be on the bus at the same time, cycle time was high (10 to 20 microseconds). Like PMOS memory, multiple power supplies were required.

The next generation of microprocessors were implemented on a 40 pin package and used N-channel metal oxide semiconductor (NMOS) technology. They had most of the features of a simple minicomputer including: direct memory addressing, interrupt facilities, unlimited subroutine nesting, separate 8-bit address and data buses and reduced interface circuitry. The higher speed of NMOS and elimination of address and data multiplexing allowed reduction in cycle time from 2 to 20 microseconds.

Bipolar transistor-transistor logic (TTL) technology is also being used to produce microcomputer building blocks. Because of TTLs high power dissipation, a complete processor cannot be put in one LSI package. Instead several packages, each containing a two of four-bit slice of the processor, are cascaded to make a complete processor of the desired word size. These types of processors are designed to be controlled by a microprogram which may be on another chip in the set, or in an external ROM. The microcycle time is 100-200 nanoseconds but since a number of microinstructions are required

to execute a program instruction, cycle time is typically one microsecond. Bit-slice microprocessors are normally used for designing general purpose computers or special purpose high speed controllers.

Bipolar I^2L technology is also being developed for use in microprocessors. It yields circuitry that is as small as MOS, as fast as TTL and consumes only 1/100 the power of TTL.

Although the architecture of microprocessors may appear primitive when compared with conventional computers, they are amazingly powerful. Since instruction execution times are on the order of two to nine microseconds, relatively fast real-time programs can be written. Furthermore, if a process requires higher speed, several processors can be used in parallel to meet the speed requirements.

IV. DATA COMPRESSION TRIALS

Several data compression trials were carried out as a first step in proving the design of the proposed flight recorder system. The DFDRs used in civilian transport aircraft record data at a constant rate of 64 12-bit words per second and are designed to retain the last 25 hours of flight data. This amounts to more than 69 million bits. At that data rate the proposed 100K solid state memory would be able to retain data for only two minutes before it is overwritten. This would not be sufficient data to determine the cause of a crash in most cases, even though the flight duration of military aircraft is much shorter than most transport flights.

A data compression method was required which would increase the retention time of significant data while still recording sufficient new data. The basic criteria used in compressing the data was to eliminate redundant information so that only data which gave significant new information about a flight would be recorded.

A 9-track computer compatible magnetic tape containing DFDR data was obtained from the NTSB Bureau of Aviation Safety, as a data base for the data compression trials. This tape contained the complete data, from takeoff to landing, of a 10.8 hour flight which encountered approximately five minutes of severe turbulence. It was felt that this data would give

a good indication of how a wide range of parameters react to the various modes and conditions of flight. The turbulent period would also give an indication of the density of data which could be expected from highly maneuverable military aircraft.

The DFDR data tape was recorded at the Bureau's flight data recorder laboratory, using a Digital Equipment Corporation (DEC) PDP 11/40 computer and was completely compatible with the PDP 11/50 computer at the Naval Postgraduate School. The PDP 11/50 facility has extensive tape handling capability, a high speed printer and a number of video display terminals for program entry and execution. The UNIX C high level language and an excellent text editor capability are also available.

Three "C" language programs were written during the process of the trials. The first, ORIGIN.C was designed to access data anywhere on the DFDR data tape, convert it to engineering units and format it for output on the terminal or printer. The printout was divided into four sections and listed 37 parameters including Greenwich Mean Time (GMT). The second program, DATA•C was designed to access data anywhere on the DFDR data tape, compress it using a simple algorithm and then store the compressed data in a file called "DFDR". The third program, RECOV•C was designed to access the compressed data in the file "DFDR", convert it to engineering units and format it for output on the terminal or printer in the same format used by ORIGIN•C. A FORTRAN

program called PLOT was also written to plot the data from ORIGIN.C and RECOV.C on the Calcomp Plotter available in the School's main computer center. The original and recovered data listings were written onto magnetic tape in IBM compatible format and used directly as the source of data for the plot routine.

The purpose of the programs ORIGIN.C and RECOV.C was to provide a second by second listing of the data, both before and after it had been compressed. The two sets of data could be easily compared at specific points using this method, but only by plotting both sets of data could the overall effect of a particular data compression method be observed. The same parameter names were used as much as possible in the four programs to make the switch from reading one program to the other as simple as possible [Table 1].

The initial attempt at data compression, DATA.C, involved use of an algorithm which compared succeeding parameter values with their most recently recorded value. If they were different, the new value was recorded along with an identifying label. Time was also recorded at the beginning of each second of data so that the time, when a particular parameter changed value, could be accurately recovered.

The labeling method used was to identify a unique number with each parameter and record it along with the parameter. Since there were more than 16 parameters considered, a minimum of 5 bits were necessary. The data values on the DFDR tape contain up to 12 significant bits, so a parameter and

GREENWICH MEAN TIME (HOURS)	GMTH
GREENWICH MEAN TIME (MINUTES)	GMTM
GREENWICH MEAN TIME (SECONDS)	GMTS
MAGNETIC HEADING	HEAD
ALTITUDE COARSE	ALTC
ALTITUDE FINE	ALTF
COMPUTED AIR SPEED	CAS
PITCH ATTITUDE	PICH
ROLL ATTITUDE	ROLL
ENG1 THRUST (N1)	ENG1
ENG2 THRUST (N1)	ENG2
ENG3 THRUST (N1)	ENG3
LONGITUDINAL ACCELERATION	LONG
VERTICAL ACCELERATION	VERG
LATERAL ACCELERATION	LATG
TOTAL AIR TEMPERATURE	TAT
HORIZONTAL STABILIZER	HSTA
ELEVATOR LEFT INBOARD	ELLI
ELEVATOR RIGHT OUTBOARD	ELRO
RUDDER POSITION, UPPER	RUDU
RUDDER POSITION, LOWER	RUDL
AILERON POSITION LEFT INBOARD	AILI
AILERON POSITION RIGHT OUTBOARD	AIRO
RIGHT HAND FLAP NO. 3	FLAP

TABLE 1. List of parameters and their corresponding variable names used throughout the programs ORIGIN.C, DATA.C, RECOV.C and PLOT.

its label would have required 17 bits. Since the PDP 11/50 has a 16-bit word, it was decided to divide each parameter which had more than six significant bits into two 6-bit parameters, coarse and fine. The letters C and F were added to the parameter names to distinguish between the coarse and fine parameters, respectively. To provide the additional label numbers an extra bit was required and the discrete parameters were packed together, six to a label. Initially 12 bits, six of data and six of label, were recorded in 16-bit words. Later to save even more memory space, four 12-bit words were packed into three 16-bit words and then recorded.

Results of the initial trial in the cruise mode indicated that, by elimination of the redundant recording of data, a compression of 7.5 to 1 could be obtained. This meant that at least 15 minutes of data could be retained using a 100K bit memory. It was found that the coarse parameters were very seldom recorded, thus providing a savings of six bits every time a parameter was recorded. Since the discrete parameters very seldom change, packing them together did not take up significant extra memory. The compression obtained using the turbulent flight data was not as good, but a reduction of 3.3 to 1 was obtained and seven minutes of data could be retained.

Careful examination of the data listings in the cruise flight mode indicated that many of the parameters were fluctuating a great deal between successive minimum change values. It was decided to modify the DATA.C program to allow

the parameters to change within specific tolerances without being recorded. The parameters were allowed to fluctuate one increment above and below their most recent recorded value. Results indicated a compression of 11.4 to 1 in the cruise mode and 3.9 to 1 in the turbulent phase.

Further tests were carried out on individual parameter, using criteria such as minimum sensor accuracy and the number of value changes during a particular time period, to determine the best tolerance. Significant increases in the compression factors were obtained, but only after the results were plotted could the overall results be observed. The data plots indicated that with greater tolerances, correspondence between the original and compressed data was not as good. This occurred because a parameter could be almost to its limit in one direction or the other (from the most recently recorded value) for an extended period of time before the change in value would finally be recorded. It was decided to modify DATA.C to include a running sum of the differences between the most recently recorded value and the successive new values. When this sum reached the parameter's tolerance, the parameter was recorded. Using this technique the tolerances could be left higher, while still retaining good correspondence between the original and recorded data.

Finally a statistical study was carried out to determine the optimum tolerance for each parameter. The assumption used was that in the cruise mode the parameter values vary within normal acceptable limits. The tolerance for each parameter was

incremented until an increase did not result in a reduction of more than one-per-minute in the number of times the parameter was recorded. Results, using tolerances arrived at with this method, indicated a compression of 14.2 to 1 in the cruise mode and 4.2 to 1 in turbulence. This would allow approximately 10 minutes of data to be retained in the turbulent phase and 30 minutes in the cruise mode.

The study was carried out on a 30 minute interval of cruise flight. The results for a 10 minute interval are contained in Tables 2 and 3, and the optimum tolerances obtained over the 30 minute interval are shown in Table 4. Figures 2 and 3 are examples of the plots of original and recovered data, obtained using the optimum tolerances. In Figure 4 the original and restored plots are superimposed,and it can be seen that they correspond very closely. The plotted data represents a three minute interval of flight and includes the first minute of turbulence. Other plots using the same interval are contained in Appendix F along with listings of the first minute of turbulent data. The final version of each of the programs is contained in Appendices B to E.

The results of the compression trials indicate that data can be effectively compressed by eliminating redundancy. By careful selection of tolerances, further compression can be obtained while still retaining good correspondence with original data.

TOLERANCE

PARAMETER	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
HEADF				68				36				28				28
ALTFF	432	292	219	180	163	144	134	125	119	111	100	104	95	93	90	84
CASF	253	168	133	113	100	90	83	79	71	67	63	56	53	52	52	50
PICHF				27				17				15				15
ROLLF				117				97				77				63
ENG1F	115	83	77	70	69	66	65	58	52	48	47	44	44	43	41	41
ENG2F	112	76	64	56	53	50	48	47	45	43	43	45	40	37	34	33
ENG3F	112	87	74	70	62	59	60	56	53	52	49	48	44	42	41	39
LONGF	340	172	128	109	93	83	66	59	60	60	53	50	44	45	43	41
VERGF				638				317				217				152
LATGF				758				335				231				177
TATF	67	41	25	23	21	19	19	16	13	11	10	11	11	11	10	10
HSTAF	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ELLIF				118				85				63				58
ELROF				30				20				18				12
RUDUF				166				136				108				100
RUFLF	358	198	137	111	93	85	71	56	51	52	45	37	36	32	31	28
AILIF				327				201				157				130
AIROF	19	11	7	5	5	5	5	3	3	3	3	3	3	3	3	3
FLAPF				1				1				1				1

TABLE 2. Table entries indicate the number of times a parameter is recorded for a particular tolerance, during a 10-minute interval in cruise flight.

TOLERANCE CHANGE

PARAMETER	1/2	2/3	3/4	4/5	5/6	6/7	7/8	8/9	9/10	10/11	11/12	12/13	13/14	14/15	15/16
HEADF							32				8				0
ALTFF	140	73	39	17	19	10	9	6	8	11	-4	9	2	3	6
CASF	85	35	20	13	10	7	4	8	4	4	7	3	1	0	2
PICHF							10				2				0
ROLLF							20				20				14
ENG1F	32	6	7	1	3	1	7	6	4	1	3	0	1	2	0
ENG2F	36	12	8	3	3	2	1	2	2	0	-2	5	3	3	1
ENG3F	25	13	4	8	3	-1	4	3	1	3	1	4	3	1	2
LONGF	168	44	19	16	10	17	7	-1	0	7	3	6	-1	2	2
VERGF							321				100				65
LATGF							423				104				54
TATF	26	16	2	2	2	0	3	3	2	1	-1	0	0	1	0
HSTAF	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ELLIF							33				22				5
ELROF							10				2				6
RUDUF							30				28				8
RUDLF	160	61	26	18	8	14	15	5	-1	7	8	1	4	1	3
AILIF							126				44				27
AIROF	8	4	2	0	0	0	2	0	0	0	0	0	0	0	0
FLAPF							0				0				0

TABLE 3. Table entries indicate the reduction in the number of times that a parameter is recorded for a particular tolerance change, during a 10 minute interval in cruise flight.

PARAMETER	TOLERANCE	TOLERANCE IN ENGINEERING UNITS	MINIMUM SENSOR ACCURACY
HEADF	8	0.70 deg	2 deg
ALTFF	5	5 feet	80 feet at 50,000
CASF	4	1 knot	4 knots at 450
PICHF	4	0.35 deg	2 deg
ROLLF	8	0.70 deg	2 deg
ENG1F	2	0.06%	2%
ENG2F	2	0.06%	2%
ENG3F	2	0.06%	2%
LONGF	4	0.002 g	0.05 g
VERGF	12	0.0275 g	0.2 g
LATGF	12	0.006 g	0.05 g
TATF	2	1°C	1°C
HSTAF	1	0.088 deg	1 deg
ELLIF	12	0.425 deg	2 deg
ELROF	4	0.15 deg	2 deg
RUDUF	12	0.39 deg	2 deg
RUDLF	4	0.13 deg	2 deg
AILIF	16	0.499 deg	2 deg
AIROF	1	0.04 deg	2 deg
FLAPF	4	0.35 deg	3 deg

TABLE 4. Table entries indicate the optimum tolerance obtained over a 30 minute interval, the corresponding tolerance in engineering units and the minimum sensor tolerance for each parameter.

Figure 2. AILERON POSITION LT-INBD (ORIGINAL) X-Scale 20 Seconds/Inch
Y-Scale 2 Deg/Inch

Figure 3. AILERON POSITION LT-INBD (RECOVERED) X-Scale 20 Seconds/Inch
Y-Scale 2 Deg/Inch

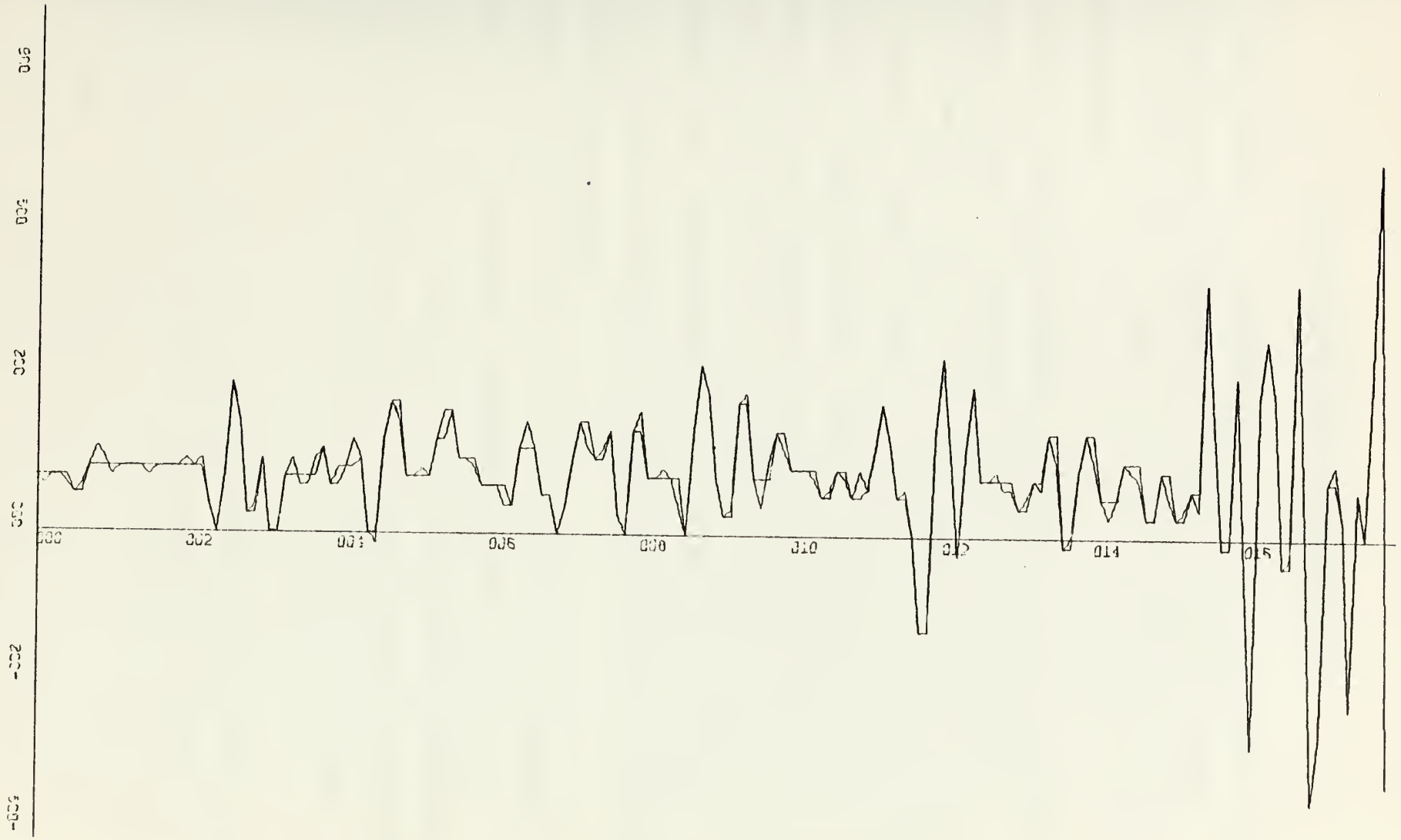


Figure 4. AILERON POSITION LT-INBD X-Scale 20 Seconds/Inch
(ORIGINAL AND RECOVERED) Y-Scale 2 Deg/Inch

V. CONCLUSIONS AND RECOMMENDATIONS

Investigation of aircraft wreckage does not provide enough investigators with adequate information. Crash-protected flight recorder data is invaluable when determining accident cause factors.

Inertial navigation systems, which are in most military aircraft, provide an excellent source of highly accurate flight parameters. Nonvolatile solid state memory is available and may be used to replace failure prone magnetic tape recording in flight recorder systems. Microprocessors are available with the capability of preprocessing flight data to conserve space for solid state memory storage. Data compression trials indicate that in cruise flight 30 minutes and in turbulent conditions 10 minutes of data can be retained in solid state memory.

These conclusions indicate that a flight data recording system using microcomputer preprocessing and nonvolatile solid state memory is feasible. It is therefore recommended that a prototype system be built and tested as soon as possible and that the inertial navigator be considered as a primary source of data.

It is hoped that this report will help develop the interest and concern required to develop this system for military aircraft and provide a starting point for further research and thesis work.

APPENDIX A

SOLID STATE MEMORY TECHNOLOGY

A. COMPUTER MEMORY TECHNOLOGY

The rapid growth of electronic data processing over the last 20 years was made possible by the development of computers using larger and faster memories. Attempts are continuing to improve the speed and reduce the cost of computer memories but memory is still one of the limiting factors in computer system development.

In the past, magnetic-ferrite core, disk, drum and tape were the dominant storage mediums, but the situation has rapidly changed since the introduction in 1970 of metal-oxide-semiconductor (MOS) memory. The first semiconductor memory applications were in cache memories and microprogram storage, which required its fast access speed. Recent developments have made them practical for main memory and may extend their application to secondary storage. Other areas of memory development that offer great promise are change-coupled devices, magnetic bubble and electron-beam-addressable memories.

The following is a short description of the various memory technologies, most applicable to future computer systems, along with the factors which influence their usefulness in flight data recording systems. MOS and bipolar are the two major semiconductor memory technologies and generally MOS provides low-cost memories while bipolar provides high-performance [Refs. 24 and 25].

1. MOS Memory Types

MOS memories are based on the metal-oxide-silicon field-effect transistor (MOSFET). Information is stored in MOS memories by depositing a charge on a parasitic or diffused capacitor and trapping it there by turning off the MOSFET. The term dynamic results from the fact that, despite the high impedence and low leakage of the MOSFET, the capacitor's charge eventually will leak off and so must be regularly refreshed to preserve the contents of memory. They are relatively slow since the low transconductance of the MOSFET reduces the amount of drive current available to charge the parasitic capacitances in the memory address and data lines. It is possible to produce static as well as dynamic MOS memories, but dynamic memories provide better speed and power characteristics than static and also allow higher chip packing density. MOS memories are volatile meaning that stored data is lost when power is removed.

a. PMOS

P-channel MOS (PMOS) is the most mature semiconductor memory technology. It got its name from the fact that the conducting channel between the source and drain of the MOSFET was p-type material which conducts holes. Multiple power supplies are required since the threshold level on PMOS memory cells is very high and a negative gate to source voltage is necessary.

b. NMOS

N-channel (NMOS) memory uses n-type material, which conducts electrons, between the source and drain. Its

operating speed is twice as fast as PMOS since electrons have a much higher carrier mobility than holes. NMOS can operate on a single 5 volt power supply since its threshold voltage is low and positive gate to source voltage is required.

c. CMOS

Complementary MOS (CMOS) memory was recently introduced and gets its name from the fact that both PMOS and NMOS devices are used in the same chip. CMOS has low power dissipation since power is consumed only during switching times when stray capacitances are charged. It is more expensive than NMOS or PMOS but can operate on a single 1.5 volt power supply making non-volatile portable memory packages practical. Higher power supply voltages are required to obtain fast memory access times.

d. SOS/MOS

Silicon-on-Sapphire MOS (SOS/MOS) memories are being developed to attack the inability of MOSFETs to quickly charge stray capacitance. Silicon is deposited on a sapphire substrate and etched to form isolated islands. Metal is poured over the insulating substrate to make interconnection and eliminates most of the stray capacitance. The resulting memory has very fast access time and requires very low power. SOS/MOS cannot be used to build dynamic memories because the silicon-sapphire interface leakage currents are too high.

2. Bipolar Memory Types

Bipolar memories are based on the expitaxial transistor. They are relatively fast due to high transconductance

but require higher power due to low input impedance. Dynamic bipolar memories are not practical due to the high leakage and low input impedance of expitaxial transistors. Like MOS, bipolar memories are volatile meaning that stored data is lost when power is removed.

a. TTL

Bipolar transistor-transistor logic (TTL) memories are input and output level compatible with TTL logic devices and require only a 5 volt power supply. Recently, Schottky diodes have been used to lower power dissipation in the disable mode but TTL memories still consume more power than MOS. Their main advantage over MOS is high access speed.

b. ECL

Bipolar emitter-coupled logic (ECL) memories, like ECL logic devices, operate in the transistors linear region. This allows extremely high speed, since the time required to bring the transistor out of saturation is eliminated, but it consumes a great deal of operating power. Like TTL, their main advantage is high access speed.

c. I^2L

Bipolar integrated-injection logic (I^2L) memories are being developed which will combine low cost with high speed and low power requirements [Ref. 26]. I^2L is a new, bipolar, large scale integration (LSI) design technique which attacks the isolation requirement of bipolar devices. By careful partitioning and judicious removal of unnecessary resistors, gates can be fabricated which do not require isolation

within the gate structure. This significantly increases chip packing density, reduces fabrication costs and increases access speed as well as reducing the required power. I^2L memories offer the low cost and power requirement of MOS and high speed of bipolar memories but have not gone into high volume production.

3. Other Memory Types

a. MNOS

Metal nitride oxide semiconductor (MNOS) memories are based on the MNOS transistor which is a MOSFET with a silicon-nitride, silicon-dioxide interface in the gate region [Refs. 18 and 25]. When a high positive voltage is applied to the gate of a MNOS memory cell a negative charge is trapped in the gate interface. This stable charge shifts the threshold of the cell to a lower voltage causing it to turn on when a low interrogation voltage is applied. If the interface was not charged the interrogation voltage would not turn on the cell.

To erase the cell a high negative voltage is applied which traps positive charges at the interface. This reverses the threshold shift effect and causes the cell to remain off when a low interrogation voltage is applied. Nonvolatility results from the fact that the trapped charges are quite stable and remain trapped even if power is removed. At present, write times are large because charge trapping requires many microseconds and the trapping mechanisms are not completely understood. Data alteration differs from conventional memories in that an erase is required prior to each write.

b. FAMOS

Floating avalanche MOS (FAMOS) memories are based on a MOSFET which has a floating silicon gate [Ref. 24]. When a high negative voltage is applied to a FAMOS memory cell, an avalanche of electrons results and is trapped on the floating gate. This lowers the threshold of the cell causing it to turn on when a low interrogation voltage is applied. If the gate was not charged, the cell would not turn on. Erasing is accomplished by neutralizing the stored negative charge with ultraviolet radiation. Like MNOS memories, FAMOS memories are nonvolatile but they have not received as much development attention.

c. CCD

Charge-coupled device (CCD) memories are based on the charge-coupled shift register [Refs. 27 to 30]. These registers are connected into loops with the output gate of each register connected to the input gate of another. One input gate in each loop is made the loop input and the previous output gate is the loop output. The basic operations involved are charge injection, charge movement, charge detection and charge regeneration. A data bit is stored in a loop by controlling the injection of an electron charge at the loop input. In order to read a particular data bit, the packets of electron charge are shifted around the loop until the correct one has reached the loop output. As each charge packet reaches the end of a register its potential is detected and then connected to the input of the next register, where the charge is regenerated.

CCD memories are dynamic since the charge packets must be refreshed periodically to ensure data retention. Refresh rate is a function of operating temperature and stored data is lost when power is removed. CCD memories are strictly serial access devices and so are block oriented rather than word or byte oriented. They are slower than MOS or bipolar memories and require more overhead circuitry, but they dissipate less power.

d. BEAMOS

Beam-addressable MOS (BEAMOS) memories are based on a new, non-structured semiconductor storage plane and an electron optical addressing component called a matrix lens [Refs. 25 and 30]. They use electron beams for accessing memory by bombarding targets of silicon in small cathode ray tubes (CRTs). They offer high density storage and extremely fast access times at low cost. Disadvantages are unproven reliability, bulky size and a high initial investment cost due to the beam deflection circuitry, special power supplies and block usage control circuitry.

e. Bubbles

Magnetic domain bubbles are the most promising magnetic technology for future memories [Refs. 25, 28, 29, and 31]. Magnetic bubbles are produced in a film of magnetic material which has been epitaxially grown. The internal magnetic field of the film tends to line up along a single "easy" axis perpendicular to its plane. Without an externally applied magnetic field, magnetization domains occupy equal "up" and "down" areas so as to minimize the total magnetic energy of

the film. If a vertical bias field of sufficient strength is applied in the "down" direction the "up" areas decrease in size to the point where they become isolated cylinders, or "up" bubbles in a "sea" of "down" magnetization.

These bubbles can be moved around with magnetic fields created by depositing islands of soft magnetic material (permalloy) on the film and applying an in-plane rotating drive field. Patterns being used for the islands are the T-bar, I-bar and a chevron. The magnetic polarities of the islands shift around in cadence with the rotating drive field and help steer the bubbles from island to island. Each rotation of the drive field makes the bubbles move one pattern.

By means of interaction between magnetization currents and the permalloy patterns it is possible to generate, expand, contract, annihilate, split, switch or detect bubbles. This versatility provides all the necessary means of memory operation. Bubble memories are nonvolatile since permanent magnets are used to provide the bias field. Bit density is very high but access time is long.

4. Memory Characteristics

TECHNOLOGY	ACCESS TIME	POWER/BIT	COST /BIT	DENSITY/CHIP
PMOS	300-400 ns	100 uW	.15¢	4K
NMOS	150-250 ns	30-100 uW	.15¢	4K
CMOS	50-250 ns	10- 20 uW	1¢	1K
SOS/MOS	10-150 ns	4- 20 uW		1K
MNOS	.8- 10 ns	150 uW	1.5¢	2K
Bipolar TTL	50- 60 ns	500 uW		1K
Bipolar ECL	45- 50 ns	500 uW		1K
Bipolar I^2L	50-100 ns		1.5¢	4K
CCD	80-250 ns	10 uW	.1¢	16K
BEAMOS	30 us	10 uW	.02¢	32M/module
Bubbles	500us-500ms		.2¢	16-100k

5. Computer Memory Forms

The computer memory technologies which have been described can be used to implement a number of different memory forms. Each has an area of application in the hierarchy of computer systems but generally as you move away from the computer the required memory is larger, slower and cheaper [Refs. 25, 29, 32 and 33].

a. RAM

Random access memory (RAM) includes all memory devices in which the contents of any address can be accessed at random in essentially the same time as any other address. RAMs may be read/write, read-only or a variation of these.

b. R/W RAM

Read/write (R/W) RAMs have the ability to store and retrieve data in about the same time, but access speed

varies with the technology being used. Most semiconductor R/W RAMs are volatile, which means that stored data is lost when power is removed. R/W RAMs are used wherever data must be frequently accessed and changed.

c. ROM

Read-only memories (ROMs) are designed to have data written into them, once only, at low speed. Each 4-bit data cell has 4 fusible nichrome or poly-silicon links, 4 fusible PN junctions or 4 laser-severable silicon islands. Information is written into the cell by selectively applying; a high current to the fusible links, a high negative voltage to the PN junctions or a minuscule laser beam to the silicon islands. After writing, the PROMs contents can be accessed at high speed and the stored data is not lost when power is removed. PROMs are used where stored information must be frequently accessed but only occasionally changed.

d. EAROM

Electrically-alterable ROMs (EAROMs) can be erased and rewritten many times. Technically they are R/W RAMs but normally they have long write times and require a separate erase step prior to writing. EAROMs are used where changes are so frequent that it is not economical to use PROMs but non-volatile random access storage is required.

e. Cyclic

Cyclic memory includes all memories in which data rotates cyclically past a read/write port [Ref. 28]. The most common cyclic memories are mechanical-type, magnetic drums and disks. Recent technological advances have resulted

in the development of electronic cyclic memories, based on magnetic bubbles and CCDs, which may eventually replace drums and disks. The disadvantage of cyclic memories is that access time, due to memory latency (the worst case access time to any random memory address), may be excessive.

Magnetic bubble memory access time can be improved by using a major/minor loop arrangement [Ref. 31]. The bubbles are rotated in minor loops and a major loop accepts the required data from the minor loops in parallel (one bit from each minor loop). The data is then read serially from the major loop. This method greatly reduces access time since the data can rotate much faster around many short loops than a single long one. Writing is carried out by serially introducing a word into the major loop and transferring it, at the proper time, to the minor loops in parallel. The goal of bubble effort is to provide a solid-state nonvolatile memory that is cheap, fast and reliable enough to provide an alternative to disk and drum storage. CCD memory access times have been improved by development of block-addressable RAM (BORAM) and line-addressable RAM (LARAM) systems [Ref. 30]. The resulting operating speeds are orders-of-magnitude faster than equivalent capacity disk memory systems but the cost per bit is greater. Since CCD memories are dynamic, refresh circuitry is required but power dissipation is very low. The aim of CCD technology is to fill the gap between high performance, high cost semiconductor memories and lower performance, lower cost disk and drum memories.

APPENDIX B

PROGRAM TO LIST ORIGINAL DATA

A. ORIGIN.C DESCRIPTION

This routine was written in UNIX C and was designed to list the original DFDR tape data in a standard format. It will access data anywhere on the DFDR data tape, convert it to engineering units and format it for output. The output may be displayed immediately on the terminal screen or directed to an output file for later display, printing, etc.

The user must determine the magnetic tape unit he is going to use (TAPEDRIVE), the first frame of DFDR data he wants to access (START) and the number of minutes of data he wants to list (LENGTH). The program must be recompiled whenever this information is changed.

The main program consists of four sections which correspond to the sections of the parameter listing which is produced. In each section the data is read from the tape, a frame at a time, and placed in a buffer array. The parameters for the section are then converted to engineering units using applicable conversion equations and formatted for output, a subframe (or second) at a time. Since the GMT data does not include seconds, a second counter is used. It is initialized at the start of each section and incremented in each subframe. Processing in a section continues until the end of the requestioned data period. The tape is then rewound to the start of the data and processing continues until all sections have been completed. Listings produced using this program can be seen in Appendix F.

B. ORIGIN.C CONVERSION EQUATIONS

The following is a list of the parameters in each section along with their conversion equations:

X is the raw data in base 10.

Y is the converted data.

GMT is calculated in each section.

GMTH If the binary value of X = 0 0|0 0 0 1|0 0|0 1 1 1, (tens | units) then Y = 17

GMTM If the binary value of X = 0 0|0 0 1 0|0 0|1 0 0 1, (tens | units) then Y = 29

GMTS The second counter is initialized at the start of each section and incremented in each subframe

SECTION I

ALTC If X $\geq$ 32, extend the sign bit to make X negative, then Y = 4096 * X

ALTF If the raw data value of ALTC was > 32, extend the sign bit to make X negative, then Y = X

CAS Y = 0.25 * X

HEAD Y = 0.087891 * X

PICH If X < 2049, then Y = 0.087891 * X
If X $\geq$ 2049, then Y = (0.087891 * X) - 360.0

ROLL If X < 2049, then Y = 0.087891 * X
If X $\geq$ 2049, then Y + (0.087891 * X) - 360.0

ENG1 Y = 0.030518 * X

ENG2 Y = 0.030518 * X

ENG3 Y = 0.030518 * X

SECTION II

LONG Y = (0.000508625 * X) - 1.0833

VERG Y = (0.0022888 * X) - 3.3755

LATG Y = (0.000508625 * X) - 1.0833

SECTION III

TAT If X $\geq$ 1024, extend the sign bit to make X negative,
then Y = 0.49951171 * X

HSTA If X < 2049, then Y = -0.938967 * X
If X $\geq$ 2049, then Y = -0.938968 * X

ELLI Y = (0.03541 * X) -72.513176

ELRO Y = (0.037474 * X) -76.7465

RUDU Y = (0.032531 * X) -66.622109

RUDL Y = (0.032531 * X) -66.622109

AILI Y =(-0.031206 * X) +63.910161

AIRO Y =(-0.0397225 * X) +81.3517

FLAP Y = (0.087891 * X) -180.0

SECTION IV (Discrete Parameters)

VHF1, VHF2

TRU1, TRD1, TRU2, TRD2, TRU3, TRD3

SL4R1, SL4R2, SL4L1, SL4L2, SL2L1, SL2L2

C. ORIGIN.C LISTING

```
#define TAPEDRIVE "/dev/mt1"
#define START 666
#define LENGTH 3
int fp1,fp2,buf[256],i,j,m,n,secnd,restart;
int gmth,gmtm,gmts,altf;
int vhf1,vhf2,tru1,trd1,tru2,trd2,tru3,trd3;
int s1411,s1412,s1211,s1212,s14r1,s14r2;
float alt,altc,cas,head,pich,roll,eng1,eng2,eng3,lng;
float elli,elro,rudl,aili,airo,tat,hsta,flap;
float verg[4];
float latg[4];
float rudu[2];

//  This routine will access data from anywhere on the
//  DFDR tape,convert it to engineering units
//  and format it for output.

//  The output may be displayed immediately on the
//  terminal or directed to an output file for
//  later display, printing, etc.

//  TAPEDRIVE indicates the magnetic tape unit to be used
//  START indicates the first frame of data
//  LENGTH indicates the number of minutes of data

main(){
fp1=open(TAPEDRIVE,0);  //  open mag tape file

//  SECTION NO. 1

secnd--;                //  initialize second counter
seek(fp1,START,4);      //  locate start frame
while(((secnd+1)/60)<LENGTH){
    read(fp1,buf,512);  //  read next data frame

    //  convert data to engineering units

    gmth=(((buf[164]&0700)>>6)*10)+(buf[164]&017);
    gmtm=(((buf[36]&0700)>>6)*10)+(buf[36]&017);
    if((buf[22]&07777)>=32)
        alt=((buf[22]&07777)|0177740)*4096.0;
    else
        alt=(buf[22]&07777)*4096.0;
    if(alt<0)
        altc=alt+4096;
    else
        altc=alt;
    eng1=(buf[32]&07777)*0.030518;
```

```
    eng2=(buf[96]&07777)*0.030518;
    eng3=(buf[160]&07777)*0.030518;
    for(i=0;i<4;i++){
        j=i*64;
        gmts=++secnd%60;     //  increment second counter
        if(alt<0)
            altf=(buf[4+j]&07777)|0170000;
        else
            altf=buf[4+j]&07777;
        cas=(buf[18+j]&07777)*0.25;
        head=(buf[2+j]&07774)*0.087891;
        if((buf[50+j]&07774)>=2049)
            pich=((buf[50+j]&07774)*0.087891)-360;
        else
            pich=(buf[50+j]&07774)*0.087891;
        if((buf[16+j]&07774)>=2049)
            roll=((buf[16+j]&07774)*0.087891)-360;
        else
            roll=(buf[16+j]&07774)*0.087891;

        //  print heading at beginning of each minute

        if((gmts%60)==0){
            printf("\n  GMT      ALTC    ALTF    CAS      ");
            printf("HEAD      PITCH");
            printf("      ROLL N1ENG1 N1ENG2 N1ENG3\n\n");
            }

        //  print data listing each second

        printf("%2d:%2d:%2d",gmth,gmtm,gmts);
        if((gmts%4)==0)
            printf("%7.0f%6d%8.2f%8.2f%9.3f%9.2f%7.2f\n",
            altc,altf,cas,head,pich,roll,eng1);
        if((gmts%4)==1)
            printf("%13d%8.2f%8.2f%9.3f%9.2f%14.2f\n",
            altf,cas,head,pich,roll,eng2);
        if((gmts%4)==2)
            printf("%13d%8.2f%8.2f%9.3f%9.2f%21.2f\n",
            altf,cas,head,pich,roll,eng3);
        if((gmts%4)==3)
            printf("%13d%8.2f%8.2f%9.3f%9.2f\n",
            altf,cas,head,pich,roll);
        }
    }

//  SECTION NO. 2

secnd= -1;  //  initialize second counter
restart=-15*LENGTH;
```

```
seek(fp1,restart,4);     //  rewind to start  frame
while(((secnd+1)/60)<LENGTH){
    read(fp1,buf,512);  //  read next data frame

    //  convert data to engineering units

    gmth=(((buf[164]&0700)>>6)*10)+(buf[164]&017);
    gmtm=(((buf[36]&0700)>>6)*10)+(buf[36]&017);
    for(i=0;i<4;i++){
        j=i*64;
        gmts=++secnd%60;     //  increment second counter
        lng=((buf[43+j]&07777)*0.000508625)-1.0833;
        for(m=0;m<4;m++){
            n=m*16;
            verg[m]=((buf[12+n+j]&07774)*0.0022888)-3.3755;
            latg[m]=((buf[14+n+j]&07774)*0.000508625)-1.0833;
            }

        //  print heading at beginning of each minute

        if((gmts%60)==0){
            printf("\n  GMT       LONG     VERG     VERG");
            printf("     VERG     VERG     LATG     LATG");
            printf("     LATG     LATG\n\n");
            }

        //  print data listing each second

        printf("%2d:%2d:%2d%8.4f",gmth,gmtm,gmts,lng);
        for(m=0;m<4;m++)
            printf("%8.4f",verg[m]);
        for(m=0;m<4;m++)
            printf("%8.4f",latg[m]);
        printf("\n");
        }
    }

//  SECTION NO. 3

secnd= -1;  //  initialize second counter
restart=-15*LENGTH;
seek(fp1,restart,4);     //  rewind to start  frame
while(((secnd+1)/60)<LENGTH){
    read(fp1,buf,512);  //  read next data frame

    //  convert data to engineering units

    gmth=(((buf[164]&0700)>>6)*10)+(buf[164]&017);
    gmtm=(((buf[36]&0700)>>6)*10)+(buf[36]&017);
    for(i=0;i<4;i++){
```

```
        j=i*64;
        gmts=++secnd%60;     //  increment second counter
        elli=((buf[40+j]&07774)*0.03541)-72.513176;
        elro=((buf[7+j]&07774)*0.037474)-76.7465;
        rudl=((buf[11+j]&07777)*0.032531)-66.622109;
        aili=((buf[8+j]&07774)*(-0.031206))+63.910161;
        airo=((buf[39+j]&07777)*(-0.0397225))+81.3517;
        for(m=0;m<2;m++){
        n=m*32;
        rudu[m]=((buf[26+n+j]&07774)*0.032531)-66.622109;
        }

        //  print heading at beginning of each minute

        if((gmts%60)==0){
            printf("\n  GMT       TAT    HSTAB ELEVLI");
            printf(" ELEVRO  RUDUP  RUDUP  RUDLO");
            printf("  AILLI  AILRO FLAPR3\n\n");
            }

        //  print data listing each second

        printf("%2d:%2d:%2d",gmth,gmtm,gmts);
        if((gmts%2)==0){
            if((buf[54+j]&07777)>=2049)
                hsta=(((buf[54+j]&07777)*0.087891)-360)
                *(-0.938967);
            else
            hsta=((buf[54+j]&07777)*0.087891)*(-0.938967);
            flap=((buf[38+j]&07774)*0.087891)-180;
        printf("%15.3f%7.3f%7.3f",hsta,elli,elro);
        printf("%7.3f%7.3f%7.3f",rudu[0],rudu[1],rudl);
        printf("%7.3f%7.3f%7.3f\n",aili,airo,flap);
            }
        if((gmts%2)==1){
            if((buf[54+j]&07777)>=1024)
                tat=((buf[54+j]&07777)|0176000)*0.49951171;
            else
                tat=(buf[54+j]&07777)*0.49951171;
        printf("%8.3f%14.3f%7.3f",tat,elli,elro);
        printf("%7.3f%7.3f%7.3f",rudu[0],rudu[1],rudl);
        printf("%7.3f%7.3f\n",aili,airo);
            }
        }
    }

//  SECTION NO. 4

secnd= -1;  //  initialize second counter
restart=-15*LENGTH;
```

```
seek(fp1,restart,4);     //  rewind to start  frame
while(((secnd+1)/60)<LENGTH){
    read(fp1,buf,512);  //  read next data frame

    //  convert data to engineering units

    gmth=(((buf[164]&0700)>>6)*10)+(buf[164]&017);
    gmtm=(((buf[36]&0700)>>6)*10)+(buf[36]&017);
    for(i=0;i<4;i++){
        j=i*64;
        gmts=++secnd%60;     //  initialize second counter
        vhf1=(buf[8+j]&02)>>1;
        vhf2=buf[8+j]&01;
        s14r2=(buf[16+j]&02)>>1;
        s14r1=buf[16+j]&01;
        s1411=(buf[28+j]&02)>>1;
        s1412=buf[28+j]&01;
        s1211=(buf[40+j]&02)>>1;
        s1212=(buf[44+j]&02)>>1;

        //  print heading at beginning of each minute

        if((gmts%60)==0){
        printf("\n  GMT     VHF1 VHF2 TRU1 TRD1 TRU2 ");
        printf("TRD2 TRU3 TRD3 S4L1 S4L2 S2L1 S2L2");
        printf(" S4R1 S4R2\n\n");
            }

        //  print data listing each second

        printf("%2d:%2d:%2d%5d%5d",gmth,gmtm,gmts,vhf1,vhf2);
        if((gmts%4)==0){
            tru1=(buf[6]&02)>>1;
            trd1=buf[6]&01;
            printf("%5d%5d%25d%5d%5d%5d%5d%5d\n",
            tru1,trd1,s1411,s1412,s1211,s1212,s14r1,s14r2);
            }
        if((gmts%4)==1){
            tru2=(buf[70]&02)>>1;
            trd2=buf[70]&01;
            printf("%15d%5d%15d%5d%5d%5d%5d%5d\n",
            tru2,trd2,s1411,s1412,s1211,s1212,s14r1,s14r2);
            }
        if((gmts%4)==2){
            tru3=(buf[134]&02)>>1;
            trd3=buf[134]&01;
            printf("%25d%5d%5d%5d%5d%5d%5d%5d\n",
            tru3,trd3,s1411,s1412,s1211,s1212,s14r1,s14r2);
            }
        if((gmts%4)==3){
```

```
                printf("%35d%5d%5d%5d%5d%5d\n",
                s1411,s1412,s1211,s1212,s14r1,s14r2);
                }
            }
        }
}
```

APPENDIX C

PROGRAM TO COMPRESS DATA

A. DATA.C DESCRIPTION

This routine was written in UNIX C and was designed to access data anywhere on the DFDR data tape, compress it using a simple algorithm and then store the compressed data in a file called "DFDR". By comparing the size (number of bits) of the "DFDR" file to the tape file, the amount of compression can be determined. For example: If the compressed file required 7188 8-bit bytes to record 10 minutes of data the compression factor would be

$$\frac{12 \times 64 \times 60 \times 10}{7188 \times 8} = 8.01$$

The user must determine the magnetic tape unit he is going to use (TAPEDRIVE) and the program must be recompiled whenever this information is changed. The program will prompt the user on the terminal for the number of the first frame of DFDR data he wants to access (START) and the number of minutes of data he wants to compress (LENGTH).

The main program consists of four sections which correspong to the four subframes of data contained in each frame [Ref. 7]. The data is read from the tape, one frame at a time, and placed in a buffer array. The parameters are processed in the order they were recorded. If a parameter occurs more than once in a frame a subroutine is called to carry out its processing. These subroutines are identified with a "W"

followed by an octal code number which corresponds to the parameter concerned.

Each parameter is processed using the following procedure. The applicable bits for the parameter are removed from its 12-bit data word and if it contains more than six significant bits it is divided into two parameters. This is the case for all parameters except the discretes and Coarse Altitude (ALTC). The upper six bits are referred to as the coarse parameter and the lower six as the fine. Their parameter names are suffixed with a "C" and "F", respectively. If the coarse parameter's value is different from its last stored value the new value is stored along with its applicable octal label. The value of the fine parameter is then compared with its last stored value and if the absolute value of their difference is greater than a predetermined tolerance, the new value is stored along with its applicable octal label. The fine parameter is also always stored when the coarse parameter changes value. The sum of the differences between the succeeding new parameter values and the parameter's last stored value is recalculated. If the absolute value of this sum is greater than the tolerance the new value is recorded.

Discrete parameters are packed into 6-bit groups. If a discrete parameters new value is different from its last recorded value, the group is recorded along with its applicable octal label.

Whenever a parameter is stored, it is placed in the lower 6 bits of a 12-bit word. The upper six bits are a 6-bit octal

label which is used to identify the parameter in memory. This is the same number which is used in identifying the subroutines. When four 12-bit parameter words are available for storage, they are packed into three 16-bit memory words and transferred to the file "DFDR" using the subroutine "Rite". Since GMT does not include seconds, a second counter is initialized whenever the minute parameter changes and is incremented and recorded at the start of each subframe.

B. DATA.C PARAMETERS

The following is a list of parameters along with their location in the 64 12-bit word DFDR data frame. Also shown are the octal labels and code names for each parameter along with the corresponding data word bits.

PARAMETER NAME	BITS	WORD	SUBFRAMES	OCTAL LABEL	CODE NAME	BITS
GMT (HOURS)	1-12	37	3	00	GMTH	1-12
GMT (MINUTES)	1-12	37	1	01	GMTM	1-12
MAGNETIC HEADING	3-12	3	1,2,3,4	03	HEADC	7-12
				04	HEADF	3-6
ALTITUDE COARSE	1-12	23	1	05	ALTC	1-6
ALTITUDE FINE	1-12	5	1,2,3,4	06	ALTF	7-12
				07	ALTFF	1-6
COMPUTED AIR SPEED	1-12	19	1,2,3,4	10	CASC	7-12
				11	CASF	1-6
PITCH ATTITUDE	3-12	51	1,2,3,4	12	PICHC	7-12
				13	PICHF	3-6
ROLL ATTITUDE	3-12	17	1,2,3,4	14	ROLLC	7-12
				15	ROLLF	3-6
ENG1 THRUST (N1)	3-12	33	1	16	ENG1C	7-12
				17	ENG1F	1-6
ENG2 THRUST (N1)	1-12	33	2	20	ENG2C	7-12
				21	ENG2F	1-6
ENG3 THRUST (N1)	1-12	33	3	22	ENG3C	7-12
				23	ENG3F	1-6
LONGITUDINAL ACCEL	1-12	44	1,2,3,4	24	LONGC	7-12
				25	LONGF	1-6

PARAMETER NAME	BITS	WORD	SUBFRAMES	OCTAL LABEL	CODE NAME	BITS
VERTICAL ACCEL	3-12	13	1,2,3,4	26	VERGC	7-12
				60	VERGF	3-6
	3-12	29	1,2,3,4	26	VERGC	7-12
				61	VERGF	3-6
	3-12	45	1,2,3,4	26	VERGC	7-12
				62	VERGF	3-6
	3-12	61	1,2,3,4	26	VERGC	7-12
				63	VERGF	3-6
LATERAL ACCEL	3-12	15	1,2,3,4	30	LATGC	7-12
				64	LATGF	3-6
	3-12	31	1,2,3,4	30	LATGC	7-12
				65	LATGF	3-6
	3-12	47	1,2,3,4	30	LATGC	7-12
				66	LATGF	3-6
	3-12	63	1,2,3,4	30	LATGC	7-12
				67	LATGF	3-6
TOTAL AIR TEMP	1-12	55	2,4	32	TATC	7-12
				33	TATF	1-6
HORIZONTAL STABILIZER	1-12	55	1,3	34	HSTAC	7-12
				35	HSTAF	1-6

PARAMETER NAME	BITS	WORD	SUBFRAMES	OCTAL LABEL	CODE NAME	BITS
ELEVATOR LT-INBD	3-12	41	1,2,3,4	36	ELLIC	7-12
				37	ELLIF	3-6
ELEVATOR RT-OUTBD	3-12	8	1,2,3,4	40	ELROC	7-12
				41	ELROF	3-6
RUDDER POSITION, UPPER	3-12	27	1,2,3,4	42	RUDUC	7-12
				43	RUDUF	3-6
	3-12	59	1,2,3,4	42	RUDUC	7-12
				57	RUDUF	3-6
RUDDER POSITION, LOWER	1-12	12	1,2,3,4	44	RUDLC	7-12
				45	RUDLF	1-6
AILERON POSIT LT-INBD	3-12	9	1,2,3,4	46	AILIC	7-12
				47	AILIF	3-6
AILERON POSIT RT-OUTBD	1-12	40	1,2,3,4	50	AIROC	7-12
				51	AIROF	1-6
RIGHT HAND FLAP NO.3	3-12	39	1,3	52	FLAPC	7-12
				53	FLAPF	3-6

PARAMETER NAME	BITS	WORD	SUBFRAMES	OCTAL LABEL	CODE NAME
VHF 1 KEYING	1	9	1,2,3,4	54	VHF1
VHF 2 KEYING	2	9	1,2,3,4	54	VHF2
ENG1 THRUST REV UNLOCK	1	7	1	55	TRU1
ENG1 THRUST REV DEPLOY	2	7	1	55	TRD1
ENG2 THRUST REV UNLOCK	1	7	2	55	TRU2
ENG2 THRUST REV DEPLOY	2	7	2	55	TRD2
ENG3 THRUST REV UNLOCK	1	7	3	55	TRU3
ENG3 THRUST REV DEPLOY	2	7	3	55	TRD3
SLAT 4 RT-OUTBD (MSB)	1	17	1,2,3,4	56	SL4R1
SLAT 4 RT-OUTBD (LSB)	2	17	1,2,3,4	56	SL4R2
SLAT 4 LT-OUTBD (LSB)	1	29	1,2,3,4	56	SL4L2
SLAT 4-LT-OUTBD (MSB)	2	29	1,2,3,4	56	SL4L1
SLAT 2 LT-INBD (MSB)	2	41	1,2,3,4	56	SL2L1
SLAT 2 LT-INBD (LSB)	2	45	1,2,3,4	56	SL2L2

VHF1 and VHF2 are combined in the program as variable VHF12 and are stored in the lower two bits of a word with the octal label 54. TRU1, TRD1, TRU2, TRD2, TRU3 and TRD3 are combined in the program as variable TRUD and are stored in the lower six bits of a word with the octal label 55.

SL4R1, SL4R2, SL4L2, SL4L1, SL2L1 and SL2L2 are combined in the program as variable SLAT and are stored in the lower six bits of a word with the octal label 56.

C. DATA.C LISTING

```
# define LENGTH 151
# define TAPEDRIVE "/dev/mt1"
int fp1,fp2,buf[256],i,temp,pack[4],n,flag,E1,E2,START;
int gmth,gmtm,gmts,headc,headf,altc,altf,altff,casc,casf;
int pichc,pichf,rollc,rollf,eng1c,eng1f,eng2c,eng2f,eng3c;
int eng3f,longc,longf,vergc,vergf,latgc,latgf,tatc,tatf;
int hstac,hstaf;
int ellic,ellif,elroc,elrof,ruduc,ruduf,rudlc,rudlf;
int ailic,ailif,airoc,airof,flapc,flapf,vhf12,trud,slat;
int n00,n01,n02,n03,n04,n05,n06,n07,n10,n11,n12,n13,n14;
int n15,n16,n17,n20,n21,n22,n23,n24,n25,n26,n27,n30,n31;
int n32,n33,n34,n35,n36,n37,n40,n41,n42,n43,n44,n45,n46;
int n47,n50,n51,n52,n53,n54,n55,n56;
int heads,altfs,cass,pichs,rolls,eng1s,eng2s,eng3s,longs;
int vergs,latgs,tats,hstas,ellis,elros,rudus,rudls,ailis;
int airos,flaps;

//  This routine will access data on the DFDR tape,
//  compress it using a simple algorithm and
//  store the compressed data in the file "dfdr".

//  TAPEDRIVE indicates the magnetic tape unit to be used
//  LENGTH indicates number of data frames to be compressed

main() {
printf("START = "); //  first data frame to be compressed
scanf("%4d\n",&START);
fp2=creat("dfdr",0600); //  create a file called "dfdr"
fp1=open(TAPEDRIVE,0);  //  open mag tape file
seek(fp1,START,4);  //  locate first data frame
for(i=1;i<LENGTH;i++){
    read(fp1,buf,512);  //  read next frame of data

//  SUBFRAME NUMBER 1

    w2();
    w3(2);
    w6(4);

    /*  thrust reverser eng1  */

    temp=buf[6]&03;
    if((trud&060)!=(temp<<4)){
        trud=(trud&017)|(temp<<4);
        pack[n]=trud|05500;
        if(++n==4) rite();
        n55++;
        }
```

```
w40(7);
w46(8);
w54(8);
w44(11);
w26(12);
w30(14);
w14(16);
w56(16);
w10(18);

/*  altitude course  */

if((buf[22]&077)!=altc){
    altc=buf[22]&077;
    pack[n]=altc|0500;
    if(++n==4) rite();
    n05++;
    }

w42(26);
w26(28);
w57(28);
w30(30);

/*  engine thrust engl  */

if(buf[32]/64!=englc){
    englc=buf[32]/64;
    pack[n]=englc|01600;
    if(++n==4) rite();
    flag=1;
    n16++;
    }
temp=(buf[32]-(englc*64));
engls=engls+temp-englf;
if(((abs(temp-englf)>=2)||(abs(engls)>=2))||(flag!=0)){
    englf=temp;                              //  0.06%
    pack[n]=englf|01700;
    if(++n==4) rite();
    flag=0;
    engls=0;
    n17++;
    }

/*  greenwich mean time minutes  */

temp=(((buf[36]&0700)>>6)*10)+(buf[36]&017);
if(temp!=gmtm){
    gmtm=temp;
```

```
        pack[n]=gmtm|0100;
        if(++n==4) rite();
        gmts=0;
        n01++;
        }

    w52(38);
    w50(39);
    w36(40);
    w58(40);
    w24(43);
    w26(44);
    w59(44);
    w30(46);
    w12(50);
    w34(54);
    w42(58);
    w26(60);
    w30(62);

//  SUBFRAME NUMBER 2

    w2();
    w3(66);
    w6(68);

    /*  thrust reverser eng2  */

    temp=buf[70]&03;
    if((trud&014)!=(temp<<2)){
        trud=(trud&063)|(temp<<2);
        pack[n]=trud|05500;
        if(++n==4) rite();
        n55++;
        }

    w40(71);
    w46(72);
    w54(72);
    w44(75);
    w26(76);
    w30(78);
    w14(80);
    w56(80);
    w10(82);
    w42(90);
    w26(92);
    w57(92);
    w30(94);
```

```
    /*  engine thrust eng2  */

    if(buf[96]/64!=eng2c){
        eng2c=buf[96]/64;
        pack[n]=eng2c|02000;
        if(++n==4) rite();
        flag=1;
        n20++;
        }
    temp=(buf[96]-(eng2c*64));
    eng2s=eng2s+temp-eng2f;
    if(((abs(temp-eng2f)>=2)||(abs(eng2s)>=2))||(flag!=0)){
        eng2f=temp;                                  //  0.06%
        pack[n]=eng2f|02100;
        if(++n==4) rite();
        flag=0;
        eng2s=0;
        n21++;
        }

    w50(103);
    w36(104);
    w58(104);
    w24(107);
    w26(108);
    w59(108);
    w30(110);
    w12(114);
    w32(118);
    w42(122);
    w26(124);
    w30(126);

//  SUBFRAME NUMBER 3

    w2();
    w3(130);
    w6(132);

    /*  thrust reverser eng3  */

    temp=buf[134]&03;
    if((trud&03)!=temp){
        trud=(trud&074)|temp;
        pack[n]=trud|05500;
        if(++n==4) rite();
        n55++;
        }

    w40(135);
```

```
w46(136);
w54(136);
w44(139);
w26(140);
w30(142);
w14(144);
w56(144);
w10(146);
w42(154);
w26(156);
w57(156);
w30(158);

/*  engine thrust eng3  */

if(buf[160]/64!=eng3c){
     eng3c=buf[160]/64;
     pack[n]=eng3c|02200;
     if(++n==4) rite();
     flag=1;
     n22++;
     }
temp=(buf[160]-(eng3c*64));
eng3s=eng3s+temp-eng3f;
if(((abs(temp-eng3f)>=2)||(abs(eng3s)>=2))||(flag!=0)){
     eng3f=temp;                                  //  0.06%
     pack[n]=eng3f|02300;
     if(++n==4) rite();
     flag=0;
     eng3s=0;
     n23++;
     }

/*  greenwich mean time hours  */

temp=(((buf[164]&0700)>>6)*10)+(buf[164]&017);
if(temp!=gmth){
     gmth=temp;
     pack[n]=gmth;
     if(++n==4) rite();
     n00++;
     }

w52(166);
w50(167);
w36(168);
w58(168);
w24(171);
w26(172);
w59(172);
```

```
    w30(174);
    w12(178);
    w34(182);
    w42(186);
    w26(188);
    w30(190);

//  SUBFRAME NUMBER 4

    w2();
    w3(194);
    w6(196);
    w40(199);
    w46(200);
    w54(200);
    w44(203);
    w26(204);
    w30(206);
    w14(208);
    w56(208);
    w10(210);
    w42(218);
    w26(220);
    w57(220);
    w30(222);
    w50(231);
    w36(232);
    w58(232);
    w24(235);
    w26(236);
    w59(236);
    w30(238);
    w12(242);
    w32(246);
    w42(250);
    w26(252);
    w30(254);
    }

//  After all frames of data have been stored the final
//  values of all parameters and the number of times
//  each was stored is added to the end of the file

    write(fp2,&gmth,188);
}

//  The following subroutines are used to process data
//  which occurs more than once in a data frame

w2(){   /*  greenwich mean time seconds  */
```

```
    pack[n]=gmts++|0200;
    if(++n==4) rite();
    n02++;
    }
w3(m){  /*  magnetic heading  */
    temp=(buf[m]&07774)/64;
    if(temp!=headc){
        headc=temp;
        pack[n]=headc|0300;
        if(++n==4) rite();
        flag=1;
        n03++;
        }
    temp=(buf[m]&07774)-(headc*64);
    heads=heads+temp-headf;
    if(((abs(temp-headf)>=8)||(abs(heads)>=8))||(flag!=0)){
        headf=temp;                              // 0.70 deg
        pack[n]=headf|0400;
        if(++n==4) rite();
        flag=0;
        heads=0;
        n04++;
        }
    }
w6(m){  /*  altitude fine  */
    if(buf[m]/64!=altf){
        altf=buf[m]/64;
        pack[n]=altf|0600;
        if(++n==4) rite();
        flag=1;
        n06++;
        }
    temp=buf[m]-(altf*64);
    altfs=altfs+temp-altff;
    if(((abs(temp-altff)>=5)||(abs(altfs)>=5))||(flag!=0)){
        altff=temp;                              // 5 feet
        pack[n]=altff|0700;
        if(++n==4) rite();
        flag=0;
        altfs=0;
        n07++;
        }
    }
w10(m){ /*  computed air speed  */
    if(buf[m]/64!=casc){
        casc=buf[m]/64;
        pack[n]=casc|01000;
        if(++n==4) rite();
        flag=1;
        n10++;
```

```
        }
    temp=(buf[m]-(casc*64));
    cass=cass+temp-casf;
    if(((abs(temp-casf)>=4)||(abs(cass)>=4))||(flag!=0)){
        casf=temp;                                  // 1 knot
        pack[n]=casf|01100;
        if(++n==4) rite();
        flag=0;
        cass=0;
        n11++;
        }
    }
w12(m){ /*  pitch attitude  */
    temp=(buf[m]&07774)/64;
    if(temp!=pichc){
        pichc=temp;
        pack[n]=pichc|01200;
        if(++n==4) rite();
        flag=1;
        n12++;
        }
    temp=(buf[m]&07774)-(pichc*64);
    pichs=pichs+temp-pichf;
    if(((abs(temp-pichf)>=4)||(abs(pichs)>=4))||(flag!=0)){
        pichf=temp;                                 // 0.35 deg
        pack[n]=pichf|01300;
        if(++n==4) rite();
        flag=0;
        pichs=0;
        n13++;
        }
    }
w14(m){ /*  roll attitude  */
    temp=(buf[m]&07774)/64;
    if(temp!=rollc){
        rollc=temp;
        pack[n]=rollc|01400;
        if(++n==4) rite();
        flag=1;
        n14++;
        }
    temp=(buf[m]&07774)-(rollc*64);
    rolls=rolls+temp-rollf;
    if(((abs(temp-rollf)>=8)||(abs(rolls)>=8))||(flag!=0)){
        rollf=temp;                                 // 0.70 deg
        pack[n]=rollf|01500;
        if(++n==4) rite();
        flag=0;
        rolls=0;
        n15++;
```

```
        }
    }
w24(m){ /*  longitudinal acceleration  */
    if(buf[m]/64!=longc){
        longc=buf[m]/64;
        pack[n]=longc|02400;
        if(++n==4) rite();
        flag=1;
        n24++;
        }
    temp=(buf[m]-(longc*64));
    longs=longs+temp-longf;
    if(((abs(temp-longf)>=4)||(abs(longs)>=4))||(flag!=0)){
        longf=temp;                                  // 0.002 g
        pack[n]=longf|02500;
        if(++n==4) rite();
        flag=0;
        longs=0;
        n25++;
        }
    }
w26(m){ /*  vertical acceleration  */
    temp=(buf[m]&07774)/64;
    if(temp!=vergc){
        vergc=temp;
        pack[n]=vergc|02600;
        if(++n==4) rite();
        flag=1;
        n26++;
        }
    temp=(buf[m]&07774)-(vergc*64);
    vergs=vergs+temp-vergf;
    if(((abs(temp-vergf)>=12)||(abs(vergs)>=12))||(flag!=0)){
        vergf=temp;                                  // 0.0275 g
        if((m%64)==12)
            pack[n]=vergf|06000;
        if((m%64)==28)
            pack[n]=vergf|06100;
        if((m%64)==44)
            pack[n]=vergf|06200;
        if((m%64)==60)
            pack[n]=vergf|06300;
        if(++n==4) rite();
        flag=0;
        vergs=0;
        n27++;
        }
    }
w30(m){ /*  lateral acceleration  */
    temo=(buf[m]&07774)/64;
```

```
    if(temp!=latgc){
        latgc=temp;
        pack[n]=latgc|03000;
        if(++n==4) rite();
        flag=1;
        n30++;
        }
    temp=(buf[m]&07774)-(latgc*64);
    latgs=latgs+temp-latgf;
    if(((abs(temp-latgf)>=12)||(abs(latgs)>=12))||(flag!=0)){
        latgf=temp;                                  // 0.006 g
        if((m%64)==14)
            pack[n]=latgf|06400;
        if((m%64)==30)
            pack[n]=latgf|06500;
        if((m%64)==46)
            pack[n]=latgf|06600;
        if((m%64)==62)
            pack[n]=latgf|06700;
        if(++n==4) rite();
        flag=0;
        latgs=0;
        n31++;
        }
    }
w32(m){ /*  total air temperature  */
    if(buf[m]/64!=tatc){
        tatc=buf[m]/64;
        pack[n]=tatc|03200;
        if(++n==4) rite();
        flag=1;
        n32++;
        }
    temp=(buf[m]-(tatc*64));
    tats=tats+temp-tatf;
    if(((abs(temp-tatf)>=2)||(abs(tats)>=2))||(flag!=0)){
        tatf=temp;                                   // 1 deg
        pack[n]=tatf|03300;
        if(++n==4) rite();
        flag=0;
        tats=0;
        n33++;
        }
    }
w34(m){ /*  horizontal stabilizer  */
    if(buf[m]/64!=hstac){
        hstac=buf[m]/64;
        pack[n]=hstac|03400;
        if(++n==4) rite();
        flag=1;
```

```
        n34++;
        }
    temp=(buf[m]-(hstac*64));
    hstas=hstas+temp-hstaf;
    if(((abs(temp-hstaf)>=1)||(abs(hstas)>=1))||(flag!=0)){
        hstaf=temp;                                  // 0.088 deg
        pack[n]=hstaf|03500;
        if(++n==4) rite();
        flag=0;
        hstas=0;
        n35++;
        }
    }
w36(m){ /*  elevator lt-inbd  */
    temp=(buf[m]&07774)/64;
    if(temp!=ellic){
        ellic=temp;
        pack[n]=ellic|03600;
        if(++n==4) rite();
        flag=1;
        n36++;
        }
    temp=(buf[m]&07774)-(ellic*64);
    ellis=ellis+temp-ellif;
    if(((abs(temp-ellif)>=12)||(abs(ellis)>=12))||(flag!=0)){
        ellif=temp;                                  // 0.425 deg
        pack[n]=ellif|03700;
        if(++n==4) rite();
        flag=0;
        ellis=0;
        n37++;
        }
    }
w40(m){ /*  elevator rt-outbd  */
    temp=(buf[m]&07774)/64;
    if(temp!=elroc){
        elroc=temp;
        pack[n]=elroc|04000;
        if(++n==4) rite();
        flag=1;
        n40++;
        }
    temp=(buf[m]&07774)-(elroc*64);
    elros=elros+temp-elrof;
    if(((abs(temp-elrof)>=4)||(abs(elros)>=4))||(flag!=0)){
        elrof=temp;                                  // 0.15 deg
        pack[n]=elrof|04100;
        if(++n==4) rite();
        flag=0;
        elros=0;
```

```
        n41++;
        }
    }
w42(m){ /*  rudder position upper  */
    temp=(buf[m]&07774)/64;
    if(temp!=ruduc){
        ruduc=temp;
        pack[n]=ruduc|04200;
        if(++n==4) rite();
        flag=1;
        n42++;
        }
    temp=(buf[m]&07774)-(ruduc*64);
    rudus=rudus+temp-ruduf;
    if(((abs(temp-ruduf)>=12)||(abs(rudus)>=12))||(flag!=0)){
        ruduf=temp;                                  // 0.39 deg
        if((m%64)==26)
            pack[n]=ruduf|04300;
        if((m%64)==58)
            pack[n]=ruduf|05700;
        if(++n==4) rite();
        flag=0;
        rudus=0;
        n43++;
        }
    }
w44(m){ /*  rudder position lower  */
    if(buf[m]/64!=rudlc){
        rudlc=buf[m]/64;
        pack[n]=rudlc|04400;
        if(++n==4) rite();
        flag=1;
        n44++;
        }
    temp=(buf[m]-(rudlc*64));
    rudls=rudls+temp-rudlf;
    if(((abs(temp-rudlf)>=4)||(abs(rudls)>=4))||(flag!=0)){
        rudlf=temp;                                  // 0.13 deg
        pack[n]=rudlf|04500;
        if(++n==4) rite();
        flag=0;
        rudls=0;
        n45++;
        }
    }
w46(m){ /*  aileron position lt-inbd  */
    temp=(buf[m]&07774)/64;
    if(temp!=ailic){
        ailic=temp;
        pack[n]=ailic|04600;
```

```
        if(++n==4) rite();
        flag=1;
        n46++;
        }
    temp=(buf[m]&07774)-(ailc*64);
    ailis=ailis+temp-ailif;
    if(((abs(temp-ailif)>=16)||(abs(ailis)>=16))||(flag!=0)){
        ailif=temp;                                  // 0.499 deg
        pack[n]=ailif|04700;
        if(++n==4) rite();
        flag=0;
        ailis=0;
        n47++;
        }
    }
w50(m){ /* aileron position rt-outbd  */
    if(buf[m]/64!=airoc){
        airoc=buf[m]/64;
        pack[n]=airoc|05000;
        if(++n==4) rite();
        flag=1;
        n50++;
        }
    temp=(buf[m]-(airoc*64));
    airos=airos+temp-airof;
    if(((abs(temp-airof)>=1)||(abs(airos)>=1))||(flag!=0)){
        airof=temp;                                  // 0.04 deg
        pack[n]=airof|05100;
        if(++n==4) rite();
        flag=0;
        airos=0;
        n51++;
        }
    }
w52(m){ /*  right hand flap no.3  */
    temp=(buf[m]&07774)/64;
    if(temp!=flapc){
        flapc=temp;
        pack[n]=flapc|05200;
        if(++n==4) rite();
        flag=1;
        n52++;
        }
    temp=(buf[m]&07774)-(flapc*64);
    flaps=flaps+temp-flapf;
    if(((abs(temp-flapf)>=4)||(abs(flaps)>=4))||(flag!=0)){
        flapf=temp;                                  // 0.35 deg
        pack[n]=flapf|05300;
        if(++n==4) rite();
        flag=0;
```

```
        flaps=0;
        n53++;
        }
    }
w54(m){ /*  vhf 1 & 2  */
    temp=buf[m]&03;
    if(temp!=vhf12){
        vhf12=temp;
        pack[n]=vhf12|05400;
        if(++n==4) rite();
        n54++;
        }
    }
w56(m){ /*  slat 4 rt-outbd  */
    temp=buf[m]&03;
    if(temp!=(slat&03)){
        slat=(slat&074)|temp;
        pack[n]=slat|05600;
        if(++n==4) rite();
        n56++;
        }
    }
w57(m){ /*  slat 4 lt-outbd  */
    temp=buf[m]&03;
    if((temp<<4)!=(slat&060)){
        slat=(slat&017)|(temp<<4);
        pack[n]=slat|05600;
        if(++n==4) rite();
        n56++;
        }
    }
w58(m){ /*  slat 2 lt-inbd  */
    temp=buf[m]&02;
    if((temp<<2)!=(slat&010)){
        slat=(slat&067)|(temp<<2);
        pack[n]=slat|05600;
        if(++n==4) rite();
        n56++;
        }
    }
w59(m){ /*  slat 2 lt-inbd  */
    temp=buf[m]&02;
    if((temp<<1)!=(slat&04)){
        slat=(slat&073)|(temp<1);
        pack[n]=slat|05600;
        if(++n==4) rite();
        n56++;
        }
    }
```

```
//  This subroutine is used to pack 4 12-bit words
//  into 3 16-bit words

rite(){
    pack[0]=(pack[0]<<4)|(pack[1]>>8);
    pack[1]=(pack[1]<<8)|(pack[2]>>4);
    pack[2]=(pack[2]<<12)|pack[3];
    write(fp2,pack,6);
    n=0;
    }
```

APPENDIX D

PROGRAM TO RECOVER AND LIST COMPRESSED DATA

A. RECOV.C DESCRIPTION

This routine was written in UNIX C and was designed to list the compressed data in a standard format. It will access the compressed data from the file "DFDR", convert it to engineering units and format it for output. The output may be displayed immediately on the terminal screen or directed to an output file for later display, printing, etc.

The user must determine the number of minutes of data he wants to list (LENGTH) and the program must be recompiled whenever this information is changed.

The main program consists of four sections which correspond to the sections of the parameter listing which is produced. The sections are divided into"cases"which perform the calculations required to recover the compressed data. At the beginning of each section the compressed data file is opened and the data is read three 16-bit words at a time. Subroutine "UNPACK" is then used to convert the three 16-bit words of data into four 12-bit words. The upper six bits of each 12-bit word is the corresponding parameter's octal label which is used to identify the "CASE" which corresponds to it.

The compressed data file is organized so that time in seconds will be recorded at the beginning of each second of data. When the second parameter (GMTS) is detected, the parameters for the section concerned are converted into engineering units and formatted for output, a subframe (or second) at a

time. The parameters which had been divided into coarse and fine, by the data compressing program DATA.C, are first added together. The conversion equations and output format are the same as were used in ORIGIN.C. If a parameter did not change during a one second interval, its previous value is used. In this way redundant data is recovered and listed.

The "second" counter is initialized at the start of each section and incremented in each subframe. It is not used to provide time in seconds since GMTS is available, but it is used to determine when the correct amount of data has been recovered.

Processing in a section continues until the end of the requested data period. The compressed data file is then re-opened and processing continues until all sections have been completed.

Listings produced using this program can be seen in Appendix F.

B. RECOV.C LISTING

```
#define LENGTH 3
int fp1,buf[4],m,i,j,para,gmth,gmtm,gmts,secnd,alt,altf;
int altff,casc,casf,headc,headf,pichc,pichf,rollc,rollf;
int eng1c,eng1f,eng2c,eng2f,eng3c,eng3f,longc;
int longf,vergc,latgc,tatc,tatf,hstac,hstaf,ellic,ellif;
int elroc,elrof,ruduc,rudlc,rudlf,ailic,ailif,airoc,airof;
int flapc,flapf,vhf1,vhf2,tru1,trd1,tru2,trd2,tru3,trd3;
int s1411,s1412,s1211,s1212,s14r1,s14r2;
float alti,altc,cas,head,pich,roll,eng1,eng2,eng3,lng;
float tat,hsta,flap,elli,elro,rudl,aili,airo;
float latg[4];
float verg[4];
float rudu[2];

//  This routine will access data from the compressed
//  data file "DFDR", convert it to engineering units
//  and format it for output.

//  The output may be displayed immediately on the
//  terminal or directed to an output file for later
//  display, printing etc.

//  LENGTH indicates the number of minutes of data

main(){

//  SECTION NO. 1

fp1=open("dfdr",0);      //  open compressed data file
secnd--;    //  initialize second counter
while((secnd/60)<LENGTH){
    read(fp1,buf,6);     //  read block of 3 16-bit words
    unpack();        //  unpack data into 4 12-bit words
    for(m=0;m<4;m++){
    para=buf[m]&07700;  //  remove parameter octal label
    switch(para){       //  use data label to select case
    case 0000:
        gmth=buf[m]&077;
        break;
    case 0100:
        gmtm=buf[m]&077;
        break;
    case 0200:
        gmts=buf[m]&077;

        // convert data to engineering units

        if(altc<0){
```

```
        alt=(altf+altff)|0170000;
        alti=altc+4096.0;
        }
    else{
        alt=altf+altff;
        alti=altc;
        }
    cas=(casc+casf)*0.25;
    head=(headc+headf)*0.087891;
    if((pichc+pichf)>=2049)
        pich=((pichc+pichf)*0.087891)-360;
    else
        pich=(pichc+pichf)*0.087891;
    if((rollc+rollf)>=2049)
        roll=((rollc+rollf)*0.087891)-360;
    else
        roll=(rollc+rollf)*0.087891;
        eng1=(eng1c+eng1f)*0.030518;
        eng2=(eng2c+eng2f)*0.030518;
        eng3=(eng3c+eng3f)*0.030518;

    //  print heading at beginning of each minute

    if((secnd%60)==){
    printf("\n  GMT      ALTC    ALTF    CAS      HEAD");
    printf("      PITCH      ROLL N1ENG1 N1ENG2 N1ENG3");
    printf("\n\n");
        }

    //  print data listing each second

    if(secnd>=0){
        printf("%2d:%2d:%2d",gmth,gmtm,gmts);
        if((gmts%4)==0)
        printf("%7.0f%6d%8.2f%8.2f%9.3f%9.2f%7.2f\n",
            alti,alt,cas,head,pich,roll,eng1);
        if((gmts%4)==1)
            printf("%13d%8.2f%8.2f%9.3f%9.2f%14.2f\n",
            alt,cas,head,pich,roll,eng2);
        if((gmts%4)==2)
            printf("%13d%8.2f%8.2f%9.3f%9.2f%21.2f\n",
            alt,cas,head,pich,roll,eng3);
        if((gmts%4)==3)
            printf("%13d%8.2f%8.2f%9.3f%9.2f\n",
            alt,cas,head,pich,roll);
        }
    secnd++;     //  increment second counter
    if((secnd/60)==LENGTH) m=4; //  look for end of LENGTH
    break;
case 0300:
```

```
        headc=(buf[m]&077)*64;
        break;
case 0400:
        headf=buf[m]&077;
        break;
case 0500:
        if((buf[m]&077)>=32)
            altc=((buf[m]&077)|0177740)*4096.0;
        else
            altc=(buf[m]&077)*4096.0;
        break;
case 0600:
        altf=(buf[m]&077)*64;
        break;
case 0700:
        altff=buf[m]&077;
        break;
case 01000:
        casc=(buf[m]&077)*64;
        break;
case 01100:
        casf=buf[m]&077;
        break;
case 01200:
        pichc=(buf[m]&077)*64;
        break;
case 01300:
        pichf=buf[m]&077;
        break;
case 01400:
        rollc=(buf[m]&077)*64;
        break;
case 01500:-
        rollf=buf[m]&077;
        break;
case 01600:
        eng1c=(buf[m]&077)*64;
        break;
case 01700:
        eng1f=buf[m]&077;
        break;
case 02000:
        eng2c=(buf[m]&077)*64;
        break;
case 02100:
        eng2f=buf[m]&077;
        break;
case 02200:
        eng3c=(buf[m]&077)*64;
        break;
```

```
    case 02300:
        eng3f=buf[m]&077;
        break;
        }
    }
    }

//  SECTION NO. 2

fp1=open("dfdr",0);      //  reopen compressed data file
secnd= -1;  //  initialize second counter
while((secnd/60)<LENGTH){
    read(fp1,buf,6);     //  read block of 3 16-bit words
    unpack();   //  unpack data into 4 12-bit words
    for(m=0;m<4;m++){
    para=buf[m]&07700;  //  remove parameter octal label
    switch(para){   //  use label to select case
    case 0000:
        gmth=buf[m]&077;
        break;
    case 0100:
        gmtm=buf[m]&077;
        break;
    case 0200:
        gmts=buf[m]&077;

        //  convert data to engineering units

        lng=((longc+longf)*0.000508625)-1.0833;

        //  print heading at beginning of each minute

        if((secnd%60)==){
    printf("\n  GMT       LONG     VERG     VERG     VERG     ");
        printf("VERG     LATG     LATG     LATG     LATG\n\n");
            }

        //  print data listing each second

        if(secnd>=0){
        printf("%2d:%2d:%2d%8.4f",gmth,gmtm,gmts,lng);
        for(j=0;j<4;j++)
            printf("%8.4f",verg[j]);
        for(j=0;j<4;j++)
            printf("%8.4f",latg[j]);
        printf("\n");
        for(j=0;j<4;j++)
            verg[j]=verg[3];
        for(j=0;j<4;j++)
            latg[j]=latg[3];
```

```
        }
        secnd++;     //  increment second counter
        if((secnd/60)==LENGTH) m=4; //  look for end of LENGTH
        break;
    case 02400:
        longc=(buf[m]&077)*64;
        break;
    case 02500:
        longf=buf[m]&077;
        break;
    case 02600:
        verac=(buf[m]&077)*64;
        break;
    case 03000:
        latgc=(buf[m]&077)*64;
        break;
    case 06000:
        for(j=0;j<4;j++)
        verg[j]=(((buf[m]&077)+verac)*0.0022888)-3.3755;
        break;
    case 06100:
        for(j=1;j<4;j++)
        verg[j]=(((buf[m]&077)+verac)*0.0022888)-3.3755;
        break;
    case 06200:
        for(j=2;j<4;j++)
        verg[j]=(((buf[m]&077)+vergc)*0.0022888)-3.3755;
        break;
    case 06300:
        verg[3]=(((buf[m]&077)+verac)*0.0022888)-3.3755;
        break;
    case 06400:
        for(j=0;j<4;j++)
        latg[j]=(((buf[m]&077)+latac)*0.000508625)-1.0833;
        break;
    case 06500:
        for(j=1;j<4;j++)
        latg[j]=(((buf[m]&077)+latac)*0.000508625)-1.0833;
        break;
    case 06600:
        for(j=2;j<4;j++)
        latg[j]=(((buf[m]&077)+latac)*0.000508625)-1.0833;
        break;
    case 06700:
        latg[3]=(((buf[m]&077)+latac)*0.000508625)-1.0833;
        break;
        }
    }
    }
```

```
//  SECTION NO. 3

fp1=open("dfdr",0); //  reopen compressed data file
secnd= -1;  //  initialize second counter
while((secnd/60)<LENGTH){
    read(fp1,buf,6);    //  read block of 3 16-bit words
    unpack();   //  unpack data into 4 12-bit words
    for(m=0;m<4;m++){
    para=buf[m]&07700;  //  remove parameter octal label
    switch(para){   //  use label to select case
    case 0000:
        gmth=buf[m]&077;
        break;
    case 0100:
        gmtm=buf[m]&077;
        break;
    case 0200:
        gmts=buf[m]&077;

        //  convert data to engineering units

        if((tatc+tatf)>=1024)
            tat=((tatc+tatf)|0176000)*0.49951171;
        else
            tat=(tatc+tatf)*0.49951171;
        if((hstac+hstaf)>=2049)
            hsta=(((hstac+hstaf)*0.087891)-360)*(-0.938967);
        else
            hsta=((hstac+hstaf)*0.087891)*(-0.938967);
        elli=((ellic+ellif)*0.03541)-72.513176;
        elro=((elroc+elrof)*0.037474)-76.7465;
        rudl=((rudlc+rudlf)*0.032531)-66.622109;
        aili=((ailic+ailif)*(-0.031206))+63.910161;
        airo=((airoc+airof)*(-0.0397225))+81.3517;
        flap=((flapc+flapf)*0.087891)-180.0;

        //  print heading at beginning of each minute

        if((secnd%60)==){
        printf("\n  GMT       TAT    HSTAB ELEVLI ELEVRO  ");
        printf("RUDUP  RUDUP  RUDLO  AILLI  AILRO FLAPR3");
        printf("\n\n");
            }

        //  print data listing each second

        if(secnd>=0){
            printf("%2d:%2d:%2d",gmth,gmtm,gmts);
            if((gmts%2)==0)
printf("%15.3f%7.3f%7.3f%7.3f%7.3f%7.3f%7.3f%7.3f%7.3f\n",
```

```
hsta,elli,elro,rudu[0],rudu[1],rudl,aili,airo,flap);
     if((gmts%2)==1)
printf("%8.3f%14.3f%7.3f%7.3f%7.3f%7.3f%7.3f%7.3f\n",
          tat,elli,elro,rudu[0],rudu[1],rudl,aili,airo);
     rudu[0]=rudu[1];
          }
     secnd++;     //  increment second counter
     if((secnd/60)==LENGTH) m=4; //  look for end of LENGTH
     break;
case 03200:
     tatc=(buf[m]&077)*64;
     break;
case 03300:
     tatf=buf[m]&077;
     break;
case 03400:
     hstac=(buf[m]&077)*64;
     break;
case 03500:
     hstaf=buf[m]&077;
     break;
case 03600:
     ellic=(buf[m]&077)*64;
     break;
case 03700:
     ellif=buf[m]&077;
     break;
case 04000:
     elroc=(buf[m]&077)*64;
     break;
case 04100:
     elrof=buf[m]&077;
     break;
case 04200:
     ruduc=(buf[m]&077)*64;
     break;
case 04300:
     rudu[0]=(((buf[m]&077)+ruduc)*0.032531)-66.622109;
     rudu[1]=rudu[0];
     break;
case 05700:
     rudu[1]=(((buf[m]&077)+ruduc)*0.032531)-66.622109;
     break;
case 04400:
     rudlc=(buf[m]&077)*64;
     break;
case 04500:
     rudlf=buf[m]&077;
     break;
case 04600:
```

```
        ailic=(buf[m]&077)*64;
        break;
    case 04700:
        ailif=buf[m]&077;
        break;
    case 05000:
        airoc=(buf[m]&077)*64;
        break;
    case 05100:
        airof=buf[m]&077;
        break;
    case 05200:
        flapc=(buf[m]&077)*64;
        break;
    case 05300:
        flapf=buf[m]&077;
        break;
        }
    }
    }

//  SECTION NO. 4

fp1=open("dfdr",0); //  reopen compressed data file
secnd= -1;  //  initialize second counter
while((secnd/60)<LENGTH){
    read(fp1,buf,6);    //  read block of 3 16-bit words
    unpack();   //  unpack data into 4 12-bit words
    for(m=0;m<4;m++){
    para=buf[m]&07700;  //  remove parameter octal label
    switch(para){   //  use label to select case
    case 0000:
        gmth=buf[m]&077;
        break;
    case 0100:
        gmtm=buf[m]&077;
        break;
    case 0200:
        gmts=buf[m]&077;

        //  print heading at beginning of each minute

        if((secnd%60)==){
    printf("\n  GMT     VHF1 VHF2 TTU1 TRD1 TRU2 TRD2");
    printf(" TRU3 TRD3 S4L1 S4L2 S2L1 S2L2 S4R1 S4R2\n\n");
        }

    //  print data listing each second

    if(secnd>=0){
```

```
    printf("%2d:%2d:%2d%5d%5d",gmth,gmtm,gmts,vhf1,vhf2);
    if((gmts%4)==0)
        printf("%5d%5d%25d%5d%5d%5d%5d%5d\n",tru1,trd1,
        s1411,s1412,s1211,s1212,s14r1,s14r2);
    if((gmts%4)==1)
        printf("%15d%5d%15d%5d%5d%5d%5d%5d\n",tru2,trd2,
        s1411,s1412,s1211,s1212,s14r1,s14r2);
    if((gmts%4)==2)
        printf("%25d%5d%5d%5d%5d%5d%5d%5d\n",tru3,trd3,
        s1411,s1412,s1211,s1212,s14r1,s14r2);
    if((gmts%4)==3)
        printf("%35d%5d%5d%5d%5d%5d\n",
        s1411,s1412,s1211,s1212,s14r1,s14r2);
        }
        secnd++;    //  increment second counter
        if((secnd/60)==LENGTH) m=4; //  look for end of LENGTH
        break;
    case 05400:
        vhf1=(buf[m]&02)>>1;
        vhf2=buf[m]&01;
        break;
    case 05500:
        tru1=(buf[m]&040)>>5;
        trd1=(buf[m]&020)>>4;
        tru2=(buf[m]&010)>>3;
        trd2=(buf[m]&04)>>2;
        tru3=(buf[m]&02)>>1;
        trd3=buf[m]&01;
        break;
    case 05600:
        s1411=(buf[m]&040)>>5;
        s1412=(buf[m]&020)>>4;
        s1211=(buf[m]&010)>>3;
        s1212=(buf[m]&04)>>2;
        s14r2=(buf[m]&02)>>1;
        s14r1=buf[m]&01;
        break;
        }
    }
    }
}

//  UNPACK DATA

unpack(){
    buf[3]=buf[2]&07777;
    buf[2]=((buf[2]>>12)&017)|((buf[1]<<4)&07760);
    buf[1]=((buf[1]>>8)&0377)|((buf[0]<<8)&07400);
    buf[0]=(buf[0]>>4)&07777;
    }
```

APPENDIX E

PROGRAM TO PLOT ORIGINAL AND COMPRESSED DATA

A. PLOT DESCRIPTION

This routine was written in FORTRAN and was designed to plot the data from ORIGIN.C and RECOV.C on the Calcomp plotter. The program consists of three sections which correspond to the first three sections of the parameter listing. The fourth section is not plotted because it contains discrete parameter data which very seldom changes. The original and recovered data listings were written onto magnetic tape in IBM compatible format and used directly as the source of data for the routine.

The parameters are first read from the tape in records which correspond to the lines in the data listings. The parameters are scaled if necessary and stored until all parameters for the required time interval have been processed.

The parameter values are plotted versus time using the subroutine DRAWP which generates control data for plotting graphs on the Calcomp plotter. DRAWP is a simplified version of the Naval Postgraduate School subroutine DRAW. The arguments required in the subroutine call are; the number of points defining the curve, an array containing the X-ordinate (Time) and an array containing the Y-ordinate (Parameter). The final two arguments are arrays used to specify optional plotting control information such as the number of curves on a graph, scaling, title, etc.

Plots made using this routine can be seen in Appendix F and Section IV, Figures 2-4.

```
C
C    B.  PLOT LISTING
C
C
C    THIS ROUTINE WILL PLOT THE DATA FROM ORIGIN.C
C    AND RECOV.C ON THE CALCOMP PLOTTER.  THE PROGRAM
C    CONSISTS OF THREE SECTIONS WHICH CORRESPOND TO THE FIRST
C    THREE SECTIONS OF THE PARAMETER LISTING.  THE FOURTH
C    SECTION IS NOT PLOTTED BECAUSE IT CONTAINS DISCRETE
C    PARAMETER DATA.
C
C    IF THE ORIGINAL AND RECOVERED DATA LISTINGS ARE WRITTEN
C    ON MAGNETIC TAPE IN IBM COMPATIBLE FORMAT THEY MAY BE
C    USED DIRECTLY AS THE SOURCE OF DATA FOR THIS PROGRAM.
C
      DIMENSION SECND(720), ALTF(180), CAS(180)
      DIMENSION HEAD(180), PITCH(180), ROLL(180)
      DIMENSION ENG(180), ENG1(180), ENG2(180), ENG3(180)
      DIMENSION LONG(180), VERG(720), LATG(720)
      DIMENSION AILI(180), AIRO(180), FLAP(180), TAT(180)
      DIMENSION HSTA(180), ELLI(180), ELRC(180), RUDD(360)
      DIMENSION RUDL(180), TAT1(180), HSTA1(180), FLAP1(180)
      INTEGER*4 ITB(12)/12*0/
      REAL*4    RTB(28)/28*0.0/
      EQUIVALENCE(TITLE,RTB(5))
      REAL*8 TITLE(12)
C
C   INITIALIZE X-ORDINATE ARRAY
C
      DO 30 I=1,720
      SECND(I)=I-1
30    CONTINUE
C
C   SECTION NO. 1
C
C   1ST SECOND
C
      READ(4,100) ALTF(1), CAS(1), HEAD(1), PITCH(1),
     1ROLL(1), ENG(1)
      ENG1(1)=ENG(1)-80.0
      ENG2(1)=ENG1(1)
      ENG3(1)=ENG1(1)
      HEAD(1)=HEAD(1)-183.0
      ALTF(1)=ALTF(1)-160.0
      CAS(1)=CAS(1)-287.0
      DO 40 I=2,60
      READ(4,200) ALTF(I), CAS(I), HEAD(I), PITCH(I),
     1ROLL(I), ENG(I)
      ENG1(I)=ENG1(I-1)
      ENG2(I)=ENG2(I-1)
      ENG3(I)=ENG3(I-1)
      HEAD(I)=HEAD(I)-183.0
      ALTF(I)=ALTF(I)-160.0
      CAS(I)=CAS(I)-287.0
      IF(I-(I/4)*4.EQ.1) ENG1(I)=ENG(I)-80.0
      IF(I-(I/4)*4.EQ.2) ENG2(I)=ENG(I)-80.0
      IF(I-(I/4)*4.EQ.3) ENG3(I)=ENG(I)-80.0
40    CONTINUE
C
C   2ND SECOND
C
      READ(4,100) ALTF(61),CAS(61),HEAD(61),PITCH(61),
     1ROLL(61), ENG(61)
      ENG1(61)=ENG(61)-80.0
      ENG2(61)=ENG2(60)
      ENG3(61)=ENG3(60)
      HEAD(61)=HEAD(61)-183.0
      ALTF(61)=ALTF(61)-160.0
      CAS(61)=CAS(61)-287.0
      DO 50 I=62,120
      READ(4,200) ALTF(I), CAS(I), HEAD(I), PITCH(I),
     1ROLL(I), ENG(I)
```

```
      ENG1(I)=ENG1(I-1)
      ENG2(I)=ENG2(I-1)
      ENG3(I)=ENG3(I-1)
      HEAD(I)=HEAD(I)-183.0
      ALTF(I)=ALTF(I)-160.0
      CAS(I)=CAS(I)-287.0
      IF(I-(I/4)*4.EQ.1) ENG1(I)=ENG(I)-80.0
      IF(I-(I/4)*4.EQ.2) ENG2(I)=ENG(I)-80.0
      IF(I-(I/4)*4.EQ.3) ENG3(I)=ENG(I)-80.0
50    CONTINUE
C
C     3RD SECOND
C
      READ(4,100) ALTF(121),CAS(121),HEAD(121),PITCH(121),
     1ROLL(121), ENG(121)
      ENG1(121)=ENG(121)-80.0
      ENG2(121)=ENG2(120)
      ENG3(121)=ENG3(120)
      HEAD(121)=HEAD(121)-183.0
      ALTF(121)=ALTF(121)-160.0
      CAS(121)=CAS(121)-287.0
      DO 55 I=122,180
      READ(4,200) ALTF(I), CAS(I), HEAD(I), PITCH(I),
     1ROLL(I), ENG(I)
      ENG1(I)=ENG1(I-1)
      ENG2(I)=ENG2(I-1)
      ENG3(I)=ENG3(I-1)
      HEAD(I)=HEAD(I)-183.0
      ALTF(I)=ALTF(I)-160.0
      CAS(I)=CAS(I)-287.0
      IF(I-(I/4)*4.EQ.1) ENG1(I)=ENG(I)-80.0
      IF(I-(I/4)*4.EQ.2) ENG2(I)=ENG(I)-80.0
      IF(I-(I/4)*4.EQ.3) ENG3(I)=ENG(I)-80.0
55    CONTINUE
C
C     SECTION NO. 2
C
C     1ST SECOND
C
      READ(4,400) LONG(1), VERG(1), VERG(2), VERG(3),
     1VERG(4), LATG(1), LATG(2), LATG(3), LATG(4)
      DO 60 I=2,60
      J=(I-1)*4
      READ(4,500) LONG(I), VERG(J+1), VERG(J+2), VERG(J+3),
     1VERG(J+4), LATG(J+1), LATG(J+2), LATG(J+3), LATG(J+4)
60    CONTINUE
C
C     2ND SECOND
C
      READ(4,400) LONG(61), VERG(241), VERG(242), VERG(243),
     1VERG(244), LATG(241), LATG(242), LATG(243), LATG(244)
      DO 70 I=62,120
      J=(I-1)*4
      READ(4,500) LONG(I), VERG(J+1), VERG(J+2), VERG(J+3),
     1VERG(J+4), LATG(J+1), LATG(J+2), LATG(J+3), LATG(J+4)
70    CONTINUE
C
C     3RD SECOND
C
      READ(4,400) LONG(121), VERG(481), VERG(482), VERG(483),
     1VERG(484), LATG(481), LATG(482), LATG(483), LATG(484)
      DO 75 I=122,180
      J=(I-1)*4
      READ(4,500) LONG(I), VERG(J+1), VERG(J+2), VERG(J+3),
     1VERG(J+4), LATG(J+1), LATG(J+2), LATG(J+3), LATG(J+4)
75    CONTINUE
C
C     SECTION NO. 3
C
C     1ST SECOND
C
      READ(4,600) HSTA(1),ELLI(1),ELRC(1),RUDU(1),RUDL(2),
     1RUDL(1), AILI(1), AIRC(1), FLAP(1)
```

```
      TAT1(1)=-4.487
      DO 80 I=2,60
      J=(I-1)*2
      READ(4,700) TAT(I),HSTA(I),ELLI(I),ELRO(I),RUDU(J+1),
     1RLDU(J+2), RUDL(I), AILI(I), AIRO(I), FLAP(I)
      TAT1(I)=TAT1(I-1)
      HSTA1(I)=HSTA1(I-1)
      FLAP1(I)=FLAP1(I-1)
      IF(I-(I/2)*2.EQ.0) TAT1(I)=TAT(I)+9.0
      IF(I-(I/2)*2.EQ.0) FLAP1(I)=FLAP(I)
      IF(I-(I/2)*2.EQ.1) HSTA1(I)=HSTA(I)
80    CONTINUE
C
C     2ND SECOND
C
      READ(4,600) HSTA(61),ELLI(61),ELRO(61),RUDU(121),
     1RLDU(122), RUDL(61), AILI(61), AIRO(61), FLAP(61)
      TAT1(61)=TAT1(60)
      DO 90 I=62,120
      J=(I-1)*2
      READ(4,700) TAT(I),HSTA(I),ELLI(I),ELRO(I),RUDU(J+1),
     1RLDU(J+2), RUDL(I), AILI(I), AIRO(I), FLAP(I)
      TAT1(I)=TAT1(I-1)
      HSTA1(I)=HSTA1(I-1)
      FLAP1(I)=FLAP1(I-1)
      IF(I-(I/2)*2.EQ.0) TAT1(I)=TAT(I)+9.0
      IF(I-(I/2)*2.EQ.0) FLAP1(I)=FLAP(I)
      IF(I-(I/2)*2.EQ.1) HSTA1(I)=HSTA(I)
90    CONTINUE
C
C     3RD SECOND
C
      READ(4,600) HSTA(121),ELLI(121),ELRO(121),RUDL(241),
     1RLDU(242), RUDL(121), AILI(121), AIRO(121), FLAP(121)
      TAT1(121)=TAT1(120)
      DO 95 I=122,180
      J=(I-1)*2
      READ(4,700) TAT(I),HSTA(I),ELLI(I),ELRO(I),RUDU(J+1),
     1RLDU(J+2), RUDL(I), AILI(I), AIRO(I), FLAP(I)
      TAT1(I)=TAT1(I-1)
      HSTA1(I)=HSTA1(I-1)
      FLAP1(I)=FLAP1(I-1)
      IF(I-(I/2)*2.EQ.0) TAT1(I)=TAT(I)+9.0
      IF(I-(I/2)*2.EQ.0) FLAP1(I)=FLAP(I)
      IF(I-(I/2)*2.EQ.1) HSTA1(I)=HSTA(I)
95    CONTINUE
100   FORMAT(///15X,F6.0,F8.2,F8.2,F9.3,F9.2,F21.2)
200   FORMAT(15X,F6.0,F8.2,F8.2,F9.3,F9.2,F21.2)
300   FORMAT(6A8)
400   FORMAT(///8X,9F8.4)
500   FORMAT(8X,9F8.4)
600   FORMAT(///8X,F15.3,8F7.3)
700   FORMAT(8X,F8.3,9F7.3)
C
C     THE PARAMETER VALUES ARE PLOTTED USING THE SUBROUTINE
C     DRAWP WHICH GENERATES CONTROL DATA
C     THE CALCOMP PLOTTER
C
      ITB(3)=9
      ITB(12)=1
      RTB(2)=20.0
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,ALTF,ITB,RTB)
      RTB(2)=2.0
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,CAS,ITB,RTB)
      RTB(2)=1.0
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,HEAD,ITB,RTB)
      RTB(2)=0.5
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,PITCH,ITB,RTB)
      RTB(2)=1.0
```

```
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,RCLL,ITB,RTB)
      RTB(2)=5.0
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,ENG1,ITB,RTB)
      RTB(2)=5.0
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,ENG2,ITB,RTB)
      RTB(2)=5.0
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,ENG3,ITB,RTB)
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,LONG,ITB,RTB)
      ITB(8)=1
      RTB(2)=0.0
      RTB(1)=80.0
      READ(5,300) TITLE
      CALL DRAWP(720,SECND,VERG,ITB,RTB)
      READ(5,300) TITLE
      CALL DRAWP(720,SECND,LATG,ITB,RTB)
      ITB(8)=0
      RTB(2)=1.0
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,TAT1,ITB,RTB)
      RTB(2)=0.5
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,ELLI,ITB,RTB)
      RTB(2)=0.2
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,ELRO,ITB,RTB)
      RTB(2)=0.5
      READ(5,300) TITLE
      CALL DRAWP(360,SECND,RUDU,ITB,RTB)
      RTB(2)=0.5
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,RUDL,ITB,RTB)
      RTB(2)=2.0
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,AILI,ITB,RTB)
      RTB(2)=0.05
      READ(5,300) TITLE
      CALL DRAWP(180,SECND,AIRC,ITB,RTB)
      STOP
      END
```

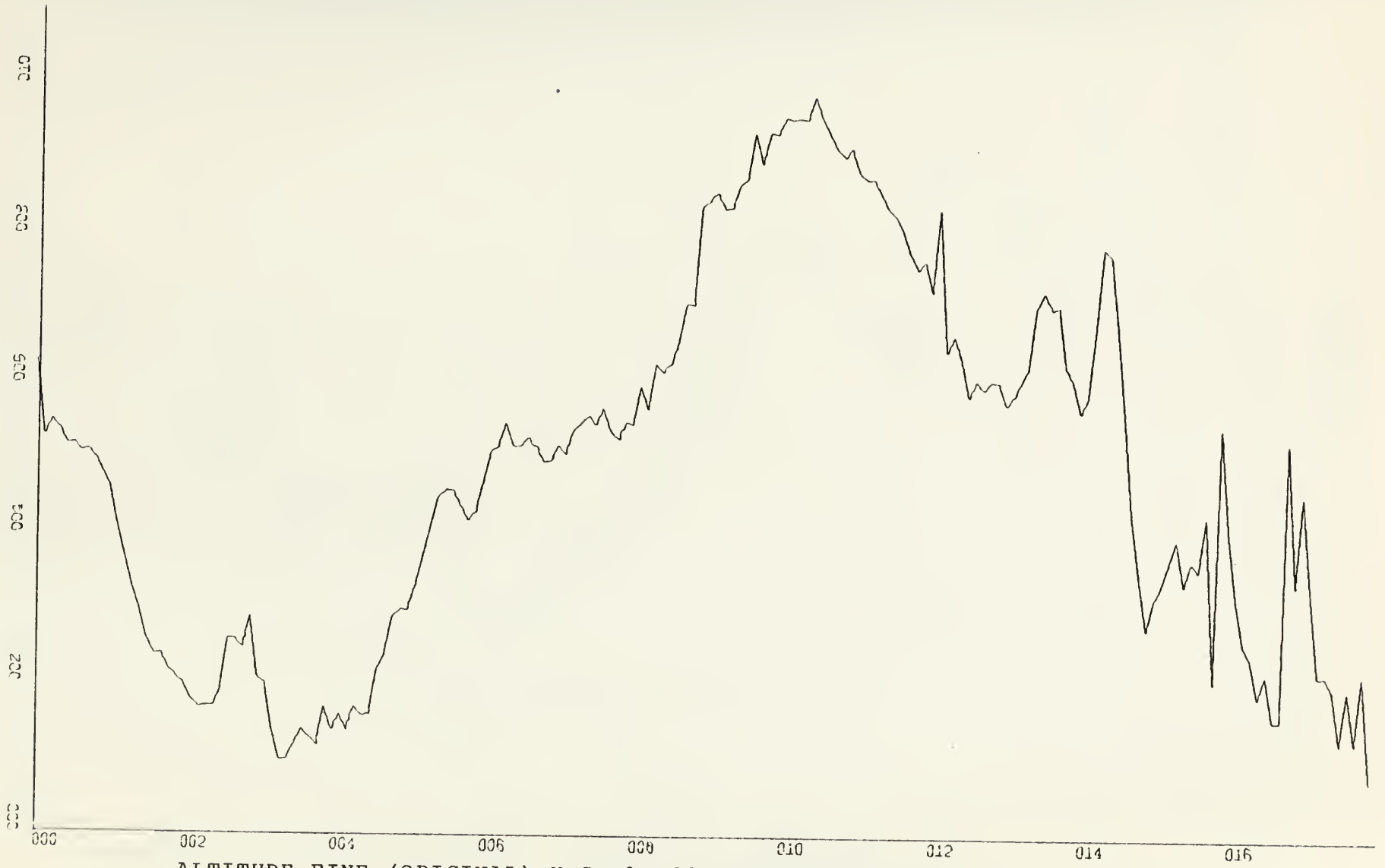
APPENDIX F

FLIGHT PARAMETER LISTINGS AND PLOTS

This appendix contains examples of listings and plots produced by the programs ORIGIN.C, RECOV.C and PLOT. The plots contain three minutes of data which includes the first minute of turbulence. The original and restored parameters; Fine Altitude, Computed Air Speed, Roll Attitude, Engine #1 Thrust and Vertical Acceleration are plotted. The recovered data plot of each parameter is followed by its original data plot for easy comparison.

The listed data corresponds to the last minute of data in the plots. The four sections of recovered data are followed by the corresponding four sections of original data.

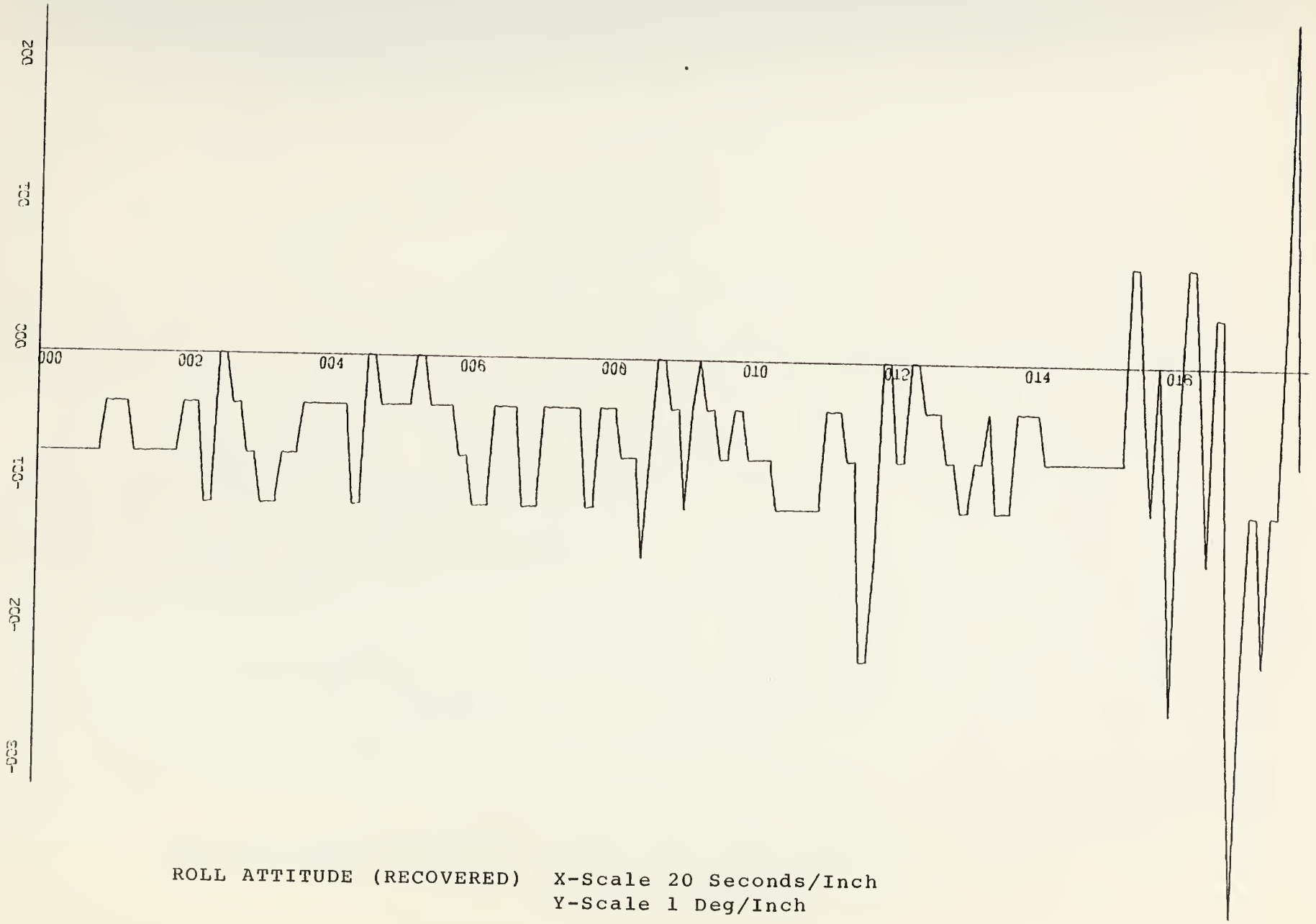
ALTITUDE FINE (RECOVERED) X-Scale 20 Seconds/Inch
Y-Scale 20 Feet/Inch Add 160 Feet to all Y values



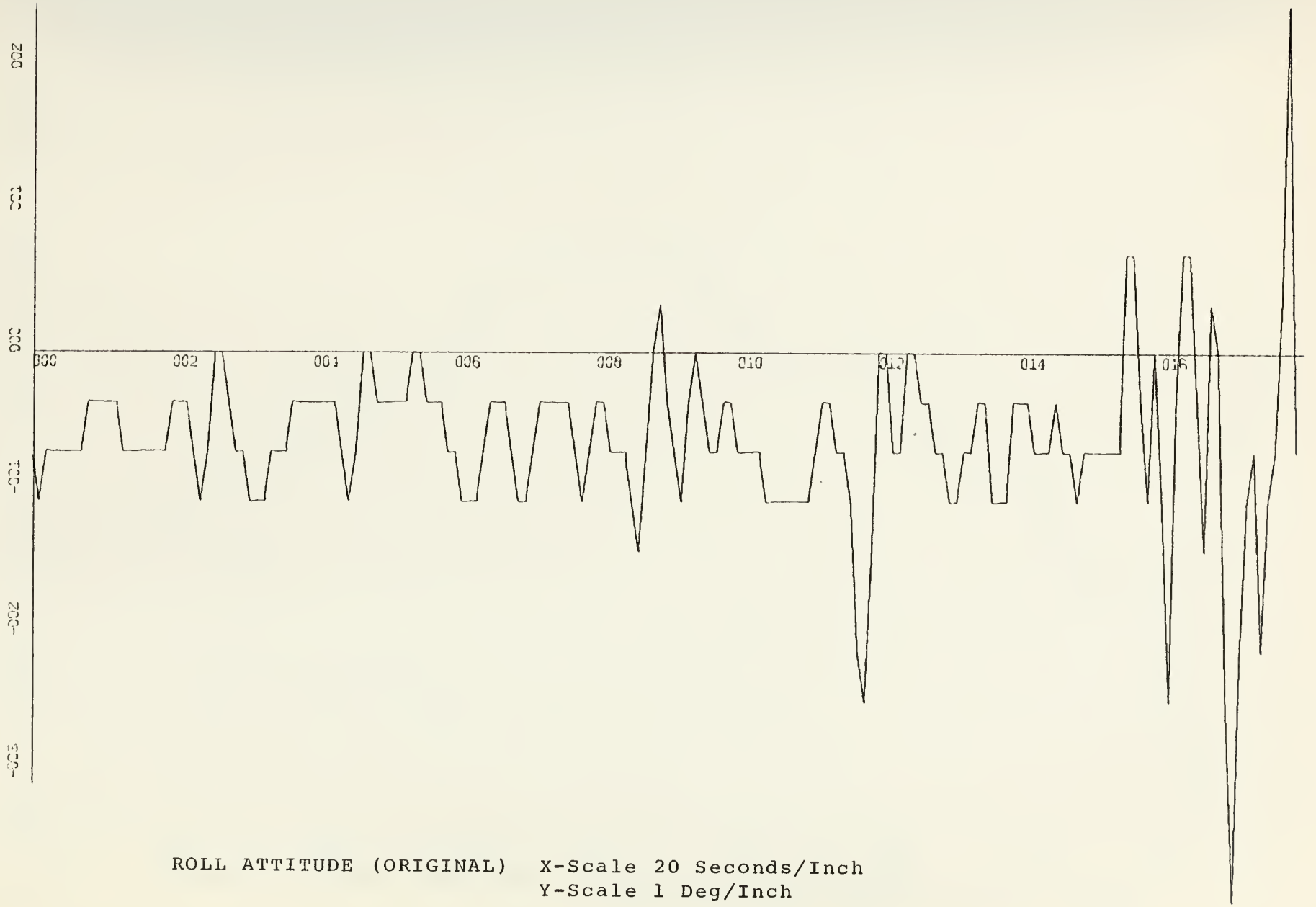
ALTITUDE FINE (ORIGINAL) X-Scale 20 Seconds/Inch
Y-Scale 20 Feet/Inch Add 160 Feet to all Y values

COMPUTED AIR SPEED (RESTORED) X-Scale 20 Seconds/Inch
Y-Scale 2 Knots/Inch Add 287 to all Y values

COMPUTED AIR SPEED (ORIGINAL) X-Scale 20 Seconds/Inch
Y-Scale 2 Knots/Inch Add 287 to all Y values

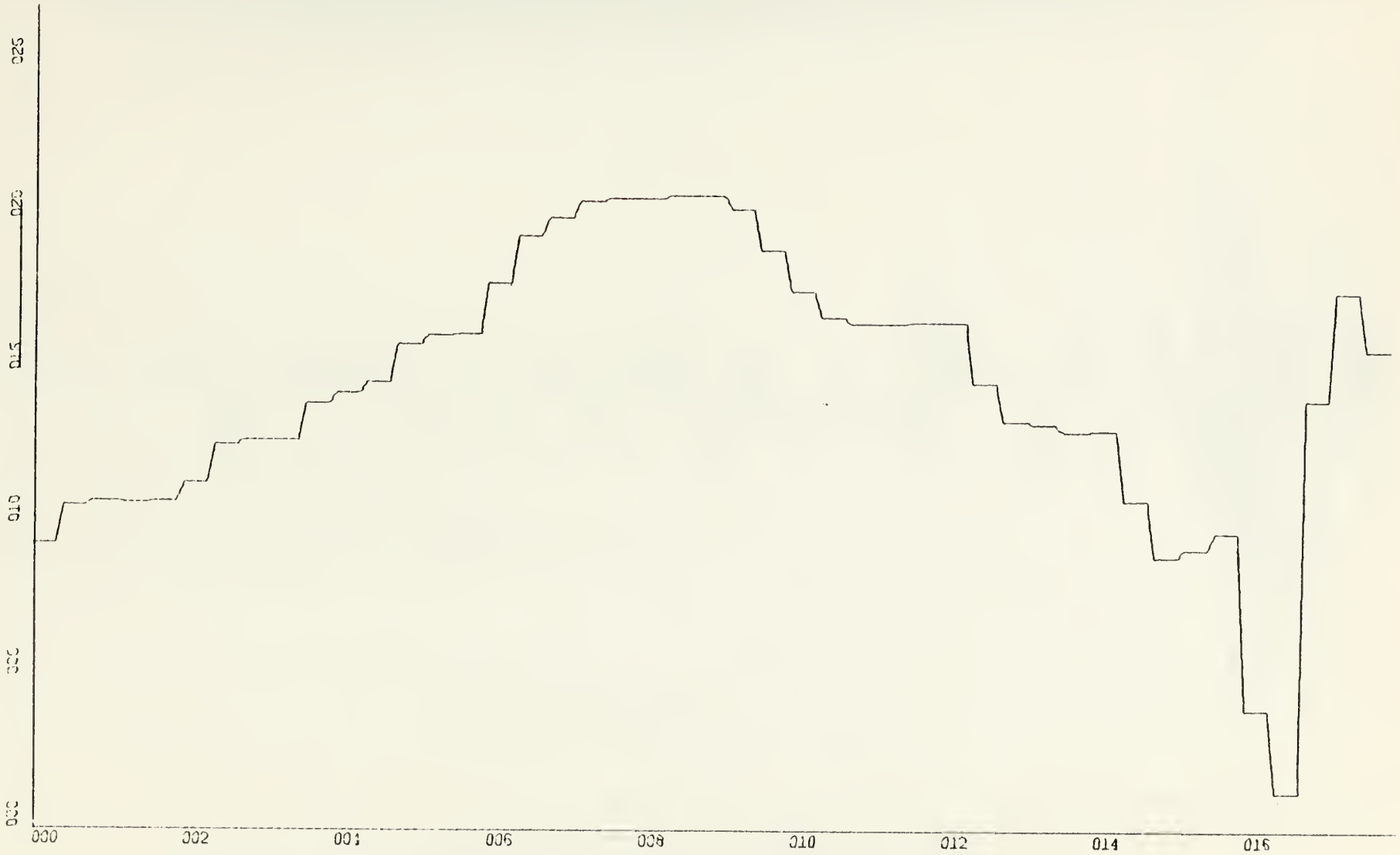


ROLL ATTITUDE (RECOVERED) X-Scale 20 Seconds/Inch
Y-Scale 1 Deg/Inch



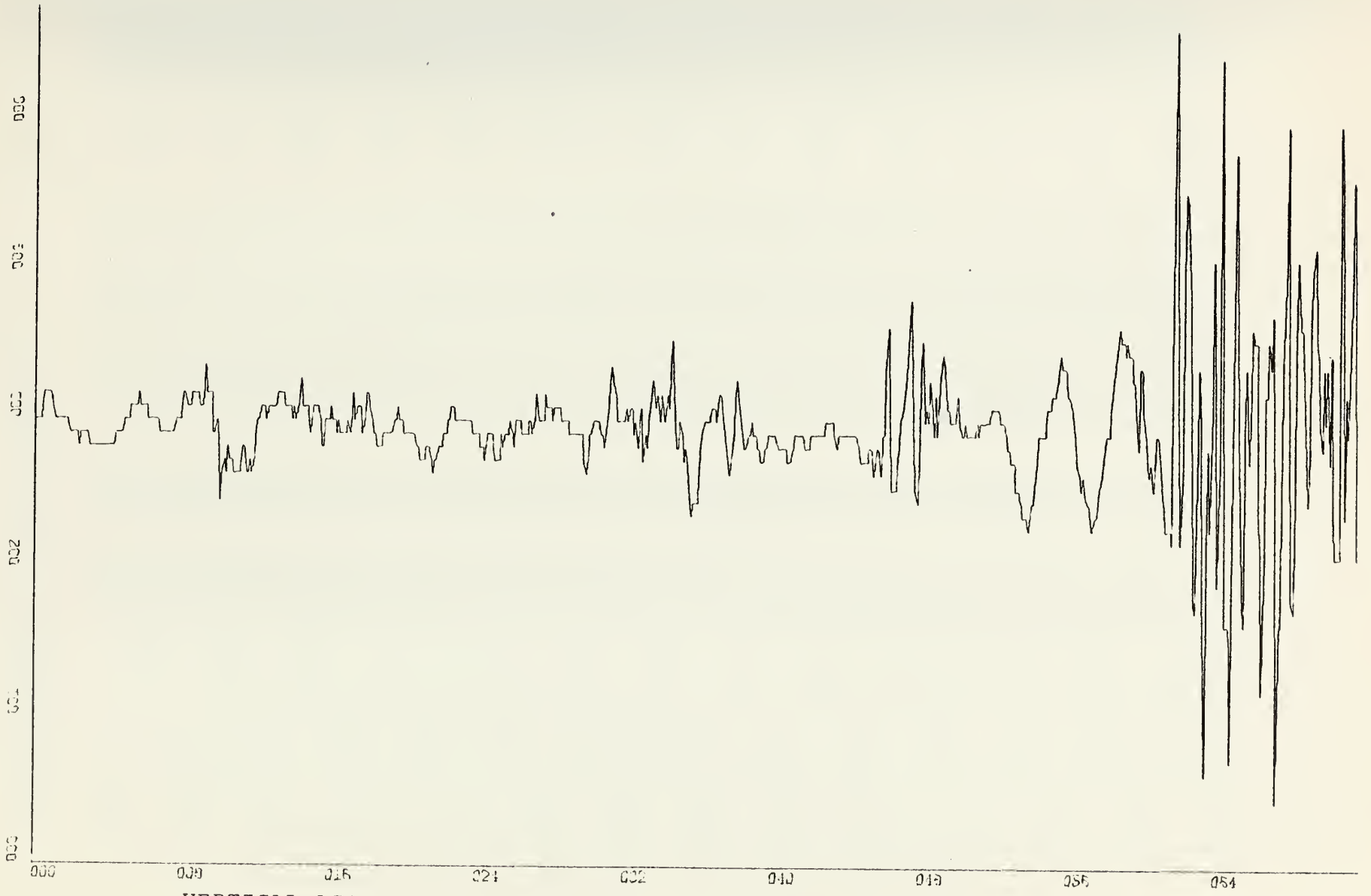
ROLL ATTITUDE (ORIGINAL) X-Scale 20 Seconds/Inch
Y-Scale 1 Deg/Inch

ENGINE #1 THRUST (RECOVERED) X-Scale 20 Seconds/Inch
Y-Scale 5%/Inch Add 80% to all Y values



ENGINE #1 THRUST (ORIGINAL) X-Scale 20 Seconds/Inch
Y-Scale 5%/Inch Add 80% to all Y values

VERTICAL ACCELERATION (RECOVERED) X-Scale 20 Seconds/Inch
Y-Scale 0.1g/Inch Add 0.7g to all Y values



VERTICAL ACCELERATION (ORIGINAL) X-Scale 20 Seconds/Inch
Y-Scale 0.1g/Inch Add 0.7g to all Y values

GMT	ALTC	ALTF	CAS	HEAD	PITCH	ROLL	N1ENG1	N1ENG2	N1ENG3
21:41: 0	32768	233	295.75	187.38	1.758	0.00	96.65		
21:41: 1		244	295.75	187.38	1.406	0.00		95.25	
21:41: 2		225	295.75	187.38	1.406	-0.70			96.01
21:41: 3		225	295.75	187.03	1.406	-0.70			
21:41: 4	32768	225	295.75	187.03	1.406	0.00	94.70		
21:41: 5		219	295.75	187.03	1.406	0.00		93.29	
21:41: 6		219	295.50	186.68	1.406	-0.35			94.15
21:41: 7		219	295.50	186.68	1.406	-0.35			
21:41: 8	32768	221	295.50	186.68	1.406	-0.35	93.42		
21:41: 9		221	294.75	186.68	1.406	-0.70		92.32	
21:41:10		221	294.75	186.68	1.406	-0.70			93.54
21:41:11		219	294.75	186.68	1.055	-1.05			
21:41:12	32768	219	294.75	186.68	1.055	-1.05	93.32		
21:41:13		223	295.50	186.68	1.055	-0.70		92.26	
21:41:14		231	295.50	186.68	1.055	-0.70			93.45
21:41:15		231	294.00	186.33	1.055	-0.35			
21:41:16	32768	231	294.00	186.33	1.055	-1.05	93.08		
21:41:17		231	293.25	186.33	1.055	-1.05		92.04	
21:41:18		223	293.25	186.68	1.055	-1.05			93.35
21:41:19		223	293.25	186.68	1.055	-0.35			
21:41:20	32768	217	294.75	186.68	1.055	-0.35	93.08		
21:41:21		217	296.00	185.98	1.055	-0.35		91.95	
21:41:22		228	296.00	185.98	1.055	-0.35			93.08
21:41:23		239	297.00	185.98	1.055	-0.70			
21:41:24	32768	239	297.00	186.33	1.055	-0.70	90.85		
21:41:25		228	295.75	186.33	1.055	-0.70		88.47	
21:41:26		216	295.75	186.33	1.406	-0.70			89.23
21:41:27		203	294.75	186.33	1.406	-0.70			
21:41:28	32768	195	293.75	186.33	1.406	-0.70	88.99		
21:41:29		188	293.75	186.33	1.406	-0.70		87.62	
21:41:30		192	293.75	186.33	1.406	-0.70			88.99
21:41:31		192	293.75	186.33	1.406	-0.70			
21:41:32	32768	197	293.75	186.33	1.406	-0.70	89.23		
21:41:33		197	293.75	186.33	1.055	-0.70		87.68	
21:41:34		197	293.75	186.33	1.406	-0.70			89.08
21:41:35		197	293.75	186.68	1.055	0.70			
21:41:36	32768	197	294.75	186.68	1.055	0.70	89.78		
21:41:37		203	295.75	185.63	0.703	-0.35		87.77	
21:41:38		181	298.00	185.63	1.055	-1.05			89.17
21:41:39		215	298.00	185.63	1.055	0.00			
21:41:40	32768	201	298.00	185.63	1.055	-1.05	83.99		
21:41:41		192	296.25	185.63	0.703	-2.46		81.03	
21:41:42		186	293.75	185.98	1.406	-0.35			81.18
21:41:43		186	293.75	185.27	1.406	0.70			
21:41:44	32768	179	292.00	184.57	1.406	0.70	81.27		
21:41:45		179	289.75	184.57	1.406	-0.35		80.63	
21:41:46		179	287.75	184.57	1.055	-1.40			86.46
21:41:47		179	289.00	184.57	1.406	0.35			
21:41:48	32768	213	289.00	183.52	0.703	0.35	94.12		
21:41:49		194	287.75	183.52	1.055	-2.46		93.75	
21:41:50		206	291.25	183.52	1.055	-3.87			97.29
21:41:51		193	291.25	184.57	1.406	-2.11			
21:41:52	32768	182	294.00	184.57	1.055	-1.05	97.69		
21:41:53		182	294.00	184.57	1.055	-1.05		95.70	
21:41:54		182	294.00	184.22	1.055	-2.11			97.11
21:41:55		173	292.25	184.22	1.055	-1.05			
21:41:56	32768	180	293.25	184.57	1.055	-1.05	95.80		
21:41:57		173	294.50	184.57	1.406	0.35		94.03	
21:41:58		182	294.50	184.57	1.055	2.46			94.58
21:41:59		168	296.25	184.57	1.055	-0.70			

SECTION NO. 1 RECOVERED DATA

GMT	LONG	VERG	VERG	VERG	VERG	LATG	LATG	LATG	LATG
21:41: 0	0.0423	0.9549	0.9824	1.0373	1.0373	0.0174	0.0174	0.0052	0.0174
21:41: 1	0.0397	1.0007	1.0007	1.0007	1.0282	0.0174	-0.0070	0.0011	0.0072
21:41: 2	0.0347	1.0098	1.0098	1.0190	0.9915	0.0072	0.0194	0.0235	0.0235
21:41: 3	0.0403	1.0282	1.0282	1.0465	1.0465	0.0336	0.0336	0.0255	0.0214
21:41: 4	0.0382	1.0098	1.0098	1.0098	1.0098	0.0214	0.0174	0.0113	0.0113
21:41: 5	0.0352	1.0007	1.0007	1.0190	1.0007	0.0113	0.0113	0.0133	0.0133
21:41: 6	0.0352	1.0007	1.0007	1.0007	0.9915	0.0133	0.0133	0.0133	0.0113
1:41: 7	0.0336	0.9915	0.9915	0.9915	0.9915	0.0113	0.0113	0.0113	0.0113
21:41: 8	0.0336	0.9915	0.9915	0.9915	1.0007	0.0113	0.0113	0.0113	0.0113
21:41: 9	0.0336	1.0007	1.0007	1.0007	1.0007	0.0113	0.0113	0.0113	0.0113
21:41:10	0.0336	1.0007	1.0007	1.0098	1.0098	0.0113	0.0133	0.0133	0.0174
21:41:11	0.0336	1.0098	1.0098	1.0098	1.0007	0.0174	0.0174	0.0113	0.0113
21:41:12	0.0336	1.0007	0.9824	0.9824	0.9824	0.0113	0.0113	0.0174	0.0174
21:41:13	0.0316	0.9824	0.9549	0.9549	0.9549	0.0174	0.0214	0.0214	0.0174
21:41:14	0.0316	0.9549	0.9366	0.9366	0.9366	0.0174	0.0133	0.0072	0.0072
21:41:15	0.0316	0.9366	0.9366	0.9366	0.9366	0.0011	0.0011	0.0011	0.0031
21:41:16	0.0316	0.9641	0.9641	0.9915	0.9915	0.0092	0.0092	0.0113	0.0113
21:41:17	0.0341	0.9915	0.9915	0.9915	1.0098	0.0113	0.0153	0.0214	0.0214
21:41:18	0.0341	1.0098	1.0190	1.0190	1.0190	0.0235	0.0194	0.0255	0.0214
21:41:19	0.0341	1.0190	1.0373	1.0373	1.0373	0.0214	0.0214	0.0194	0.0194
21:41:20	0.0341	1.0373	1.0373	1.0373	1.0190	0.0113	0.0113	0.0113	0.0113
21:41:21	0.0301	1.0098	1.0098	0.9915	0.9915	0.0113	0.0113	0.0072	0.0072
21:41:22	0.0260	0.9641	0.9641	0.9641	0.9641	0.0072	0.0072	0.0092	0.0092
21:41:23	0.0235	0.9458	0.9458	0.9458	0.9274	0.0092	0.0113	0.0113	0.0113
21:41:24	0.0219	0.9274	0.9274	0.9458	0.9458	0.0153	0.0153	0.0153	0.0153
21:41:25	0.0219	0.9641	0.9641	0.9824	0.9824	0.0133	0.0133	0.0133	0.0133
21:41:26	0.0255	0.9824	1.0098	1.0190	1.0190	0.0133	0.0133	0.0133	0.0092
21:41:27	0.0275	1.0465	1.0465	1.0648	1.0648	0.0092	0.0133	0.0133	0.0133
21:41:28	0.0275	1.0648	1.0465	1.0465	1.0465	0.0174	0.0174	0.0174	0.0174
21:41:29	0.0275	1.0465	1.0465	1.0282	1.0098	0.0133	0.0133	0.0133	0.0133
21:41:30	0.0255	0.9824	1.0373	1.0373	1.0098	0.0133	0.0133	0.0072	0.0072
21:41:31	0.0199	1.0098	0.9732	0.9732	0.9732	0.0072	0.0174	0.0174	0.0174
21:41:32	0.0250	0.9549	0.9549	0.9915	0.9915	0.0092	0.0092	0.0092	0.0052
21:41:33	0.0219	0.9915	0.9549	0.9549	0.9274	0.0052	0.0113	0.0194	0.0235
21:41:34	0.0219	0.9274	0.9274	0.9274	1.0556	0.0235	0.0214	0.0296	0.0296
21:41:35	0.0357	1.2662	1.1563	1.0007	0.9183	0.0499	0.0581	0.0418	0.0255
21:41:36	0.0270	0.9641	1.1105	1.1563	1.1563	0.0113	-0.0030	0.0031	-0.0193
21:41:37	0.0138	1.0922	0.9824	0.8817	0.8817	0.0031	0.0113	0.0255	0.0255
21:41:38	0.0174	0.9732	1.0373	1.0098	0.8725	0.0357	0.0153	0.0153	0.0255
21:41:39	0.0062	0.7627	0.9274	0.9824	0.9274	-0.0152	0.0275	-0.0091	-0.0254
21:41:40	0.0062	1.0007	1.1105	1.0373	0.8908	-0.0091	-0.0091	0.0214	0.0031
21:41:41	0.0092	0.9641	1.2479	1.1014	0.8634	0.0296	0.0296	0.0804	0.0723
21:41:42	0.0046	0.8634	0.8634	0.7718	0.9000	0.0540	0.0174	0.0052	0.0377
21:41:43	0.0250	1.0922	1.1838	1.0831	0.9824	0.0092	0.0092	0.0235	0.0031
21:41:44	0.0143	0.8817	0.8634	1.0007	1.0373	0.0031	0.0092	-0.0172	-0.0172
21:41:45	0.0214	0.9732	1.0098	1.0648	1.0648	-0.0111	0.0011	-0.0070	0.0214
21:41:46	0.0291	1.0648	0.9366	0.9366	0.8176	0.0296	0.0581	0.0581	0.0642
21:41:47	0.0408	0.9000	1.0190	1.0190	1.0556	0.0906	0.0235	0.0174	-0.0132
21:41:48	0.0387	1.0556	1.0556	0.9732	0.7443	-0.0132	-0.0132	-0.0376	-0.0315
21:41:49	0.0408	0.8542	0.8725	0.9641	1.0465	-0.0315	-0.0498	-0.0355	-0.0254
21:41:50	0.0525	1.1014	1.2021	1.0190	0.8817	0.0133	0.0316	0.0174	0.0174
21:41:51	0.0418	0.8817	0.9366	1.0648	1.1105	0.0174	0.0235	0.0357	0.0357
21:41:52	0.0458	1.0648	1.0648	1.0373	1.0007	0.0397	0.0214	0.0133	-0.0070
21:41:53	0.0428	0.9458	0.9732	1.0739	1.1014	0.0011	-0.0152	-0.0030	-0.0091
21:41:54	0.0448	1.1014	1.0556	1.0556	1.0007	0.0031	0.0255	0.0194	0.0194
21:41:55	0.0418	1.0007	1.0373	1.0007	1.0373	0.0336	0.0113	0.0113	0.0031
21:41:56	0.0316	0.9732	1.0465	0.9366	0.9091	0.0031	-0.0050	-0.0233	0.0133
21:41:57	0.0397	0.9091	0.9091	1.2021	1.1746	0.0296	0.0479	0.0601	0.0296
21:41:58	0.0357	0.9366	1.0190	0.9915	1.0190	-0.0030	0.0377	0.0052	-0.0152
21:41:59	0.0214	1.0922	1.1655	1.1014	0.9091	-0.0274	-0.0091	0.0072	0.0520

SECTION NO. 2 RECOVERED DATA

GMT	TAT	HSTAB	ELEVLI	ELEVRO	RUDUP	RUDUP	RUDLO	AILLI	AILRO	FLAPR3
21:41: 0		1.402	-0.418	0.450	1.042	1.042	0.652	2.372	-0.199	0.001
21:41: 1	-13.487		0.290	0.300	0.652	0.652	0.164	1.124	-0.199	
21:41: 2		1.402	0.290	0.450	0.001	0.001	-0.259	-0.249	-0.199	0.001
21:41: 3	-12.987		0.290	0.450	0.001	0.522	-0.389	1.124	-0.199	
21:41: 4		1.402	0.290	0.450	0.522	0.522	0.099	1.997	-0.199	0.001
21:41: 5	-12.987		0.290	0.450	0.522	0.522	0.099	0.749	-0.199	
21:41: 6		1.402	0.007	0.450	0.132	0.132	-0.194	0.749	-0.199	0.001
21:41: 7	-12.987		0.007	0.450	0.132	0.132	-0.194	0.749	-0.199	
21:41: 8		1.402	0.007	0.450	0.132	0.132	-0.356	0.749	-0.199	0.001
21:41: 9	-12.987		0.148	0.450	0.132	0.132	-0.356	0.749	-0.199	
21:41:10		1.402	0.148	0.450	0.132	0.132	-0.356	0.375	-0.199	0.001
21:41:11	-12.488		0.148	0.450	0.132	0.132	-0.356	0.375	-0.199	
21:41:12		1.402	0.148	0.450	0.262	0.262	-0.194	0.749	-0.199	0.001
21:41:13	-12.488		-0.277	0.450	0.262	0.262	-0.194	0.749	-0.199	
21:41:14		1.402	-0.277	0.300	0.262	0.392	0.001	1.373	-0.199	0.001
21:41:15	-12.488		0.148	0.300	0.392	0.001	-0.129	1.373	-0.199	
21:41:16		1.402	0.148	0.300	0.001	0.001	-0.519	-0.125	-0.199	0.001
21:41:17	-12.488		0.573	0.300	0.001	0.001	-0.519	-0.125	-0.199	
21:41:18		1.402	0.573	0.300	0.262	0.262	-0.226	0.874	-0.199	0.001
21:41:19	-12.488		0.573	0.450	0.522	0.522	0.001	1.373	-0.199	
21:41:20		1.402	0.290	0.600	0.522	0.522	0.001	1.373	-0.199	0.001
21:41:21	-12.488		-0.135	0.450	0.262	0.262	-0.064	0.500	-0.199	
21:41:22		1.402	-0.560	0.450	0.001	0.001	-0.356	0.500	-0.199	0.001
21:41:23	-12.488		-0.560	0.150	0.001	0.001	-0.487	0.500	-0.199	
21:41:24		1.402	0.007	0.300	0.001	0.262	-0.487	0.999	-0.199	0.001
21:41:25	-11.988		0.431	0.300	0.262	0.262	-0.226	0.999	-0.199	
21:41:26		1.402	0.431	0.450	0.262	0.262	-0.226	0.999	-0.199	0.001
21:41:27	-11.988		0.856	0.450	0.262	0.262	-0.226	0.250	-0.199	
21:41:28		1.402	0.856	0.450	0.132	0.132	-0.226	0.250	-0.199	0.001
21:41:29	-11.988		0.431	0.600	0.132	0.262	-0.226	0.874	-0.199	
21:41:30		1.402	0.007	0.600	0.262	0.262	-0.226	0.874	-0.199	0.001
21:41:31	-11.988		0.007	0.450	0.262	0.262	-0.226	0.250	-0.199	
21:41:32		1.402	0.007	0.450	0.132	0.132	-0.226	0.250	-0.199	0.001
21:41:33	-12.488		0.007	0.450	0.132	0.132	-0.356	0.624	-0.199	
21:41:34		1.402	0.007	0.450	0.132	0.132	-0.356	0.624	-0.199	0.001
21:41:35	-12.488		1.989	0.300	0.652	1.433	0.001	3.371	-0.238	
21:41:36		1.402	0.007	0.750	1.433	0.522	0.945	1.498	-0.199	0.001
21:41:37	-12.488		1.281	0.450	0.001	0.001	-0.161	-0.125	-0.199	
21:41:38		1.402	-0.985	0.300	0.001	0.262	-0.389	-0.125	-0.199	0.001
21:41:39	-12.488		-0.418	0.000	0.262	0.262	0.001	2.122	-0.159	
21:41:40		1.402	0.573	0.150	-0.519	-0.519	-0.844	-0.499	-0.159	0.001
21:41:41	-10.989		0.998	0.300	-0.519	0.132	-0.844	-2.746	-0.278	
21:41:42		1.402	-0.135	0.600	1.172	1.172	0.359	1.873	-0.199	0.001
21:41:43	-10.989		0.856	0.150	1.172	0.652	0.880	2.622	-0.199	
21:41:44		1.402	0.431	0.600	0.652	0.262	0.229	1.873	-0.199	0.001
21:41:45	-10.989		0.431	0.450	-0.389	-0.389	-0.552	-0.374	-0.199	
21:41:46		1.402	1.281	0.450	-0.389	0.392	-1.170	-0.374	-0.199	0.001
21:41:47	-11.489		-0.418	0.600	1.693	1.693	0.359	3.371	-0.199	
21:41:48		1.402	1.848	0.300	0.652	-0.259	0.782	0.000	-0.199	0.001
21:41:49	-11.489		-0.985	0.750	-1.040	-1.820	-1.137	-3.495	-0.159	
21:41:50		1.402	1.989	0.000	-1.820	-0.519	-2.536	-2.621	-0.278	0.001
21:41:51	-11.489		-0.418	0.750	0.001	0.001	-0.552	0.749	-0.199	
21:41:52		1.402	1.140	0.150	0.652	0.652	0.001	0.749	-0.199	0.001
21:41:53	-11.489		0.148	0.600	0.652	0.001	0.197	0.250	-0.199	
21:41:54		1.402	0.856	0.600	-0.389	0.001	-0.747	-2.247	-0.199	0.001
21:41:55	-10.989		0.856	0.450	0.001	0.392	-0.389	0.624	-0.199	
21:41:56		1.402	0.290	0.600	0.392	0.001	0.001	0.000	-0.199	0.001
21:41:57	-10.989		-0.843	0.450	0.001	1.042	-0.617	2.122	-0.119	
21:41:58		1.402	1.281	0.300	1.042	1.042	0.782	4.993	-0.199	0.001
21:41:59	-10.989		0.290	0.750	0.001	-0.779	0.132	-3.245	-0.238	

SECTION NO. 3 RECOVERED DATA

GMT	VHF1	VHF2	TTU1	TRD1	TRU2	TRD2	TRU3	TRD3	S4L1	S4L2	S2L1	S2L2	S4R1	S4R2
21:41: 0	0	0	1	1					1	0	1	0	1	0
21:41: 1	0	0			1	1			1	0	1	0	1	0
21:41: 2	0	0					1	1	1	0	1	0	1	0
21:41: 3	0	0							1	0	1	0	1	0
21:41: 4	0	0	1	1					1	0	1	0	1	0
21:41: 5	0	0			1	1			1	0	1	0	1	0
21:41: 6	0	0					1	1	1	0	1	0	1	0
21:41: 7	0	0							1	0	1	0	1	0
21:41: 8	0	0	1	1					1	0	1	0	1	0
21:41: 9	0	0			1	1			1	0	1	0	1	0
21:41:10	0	0					1	1	1	0	1	0	1	0
21:41:11	0	0							1	0	1	0	1	0
21:41:12	0	0	1	1					1	0	1	0	1	0
21:41:13	0	0			1	1			1	0	1	0	1	0
21:41:14	0	0					1	1	1	0	1	0	1	0
21:41:15	0	0							1	0	1	0	1	0
21:41:16	0	0	1	1					1	0	1	0	1	0
21:41:17	0	0			1	1			1	0	1	0	1	0
21:41:18	0	0					1	1	1	0	1	0	1	0
21:41:19	0	0							1	0	1	0	1	0
21:41:20	0	0	1	1					1	0	1	0	1	0
21:41:21	0	0			1	1			1	0	1	0	1	0
21:41:22	0	0					1	1	1	0	1	0	1	0
21:41:23	0	0							1	0	1	0	1	0
21:41:24	0	0	1	1					1	0	1	0	1	0
21:41:25	0	0			1	1			1	0	1	0	1	0
21:41:26	0	0					1	1	1	0	1	0	1	0
21:41:27	0	0							1	0	1	0	1	0
21:41:28	0	0	1	1					1	0	1	0	1	0
21:41:29	0	0			1	1			1	0	1	0	1	0
21:41:30	0	0					1	1	1	0	1	0	1	0
21:41:31	0	0							1	0	1	0	1	0
21:41:32	0	0	1	1					1	0	1	0	1	0
21:41:33	0	0			1	1			1	0	1	0	1	0
21:41:34	0	0					1	1	1	0	1	0	1	0
21:41:35	0	0							1	0	1	0	1	0
21:41:36	0	0	1	1					1	0	1	0	1	0
21:41:37	0	0			1	1			1	0	1	0	1	0
21:41:38	0	0					1	1	1	0	1	0	1	0
21:41:39	0	0							1	0	1	0	1	0
21:41:40	0	0	1	1					1	0	1	0	1	0
21:41:41	0	0			1	1			1	0	1	0	1	0
21:41:42	0	0					1	1	1	0	1	0	1	0
21:41:43	0	0							1	0	1	0	1	0
21:41:44	0	0	1	1					1	0	1	0	1	0
21:41:45	0	0			1	1			1	0	1	0	1	0
21:41:46	0	0					1	1	1	0	1	0	1	0
21:41:47	0	0							1	0	1	0	1	0
21:41:48	0	0	1	1					1	0	1	0	1	0
21:41:49	0	0			1	1			1	0	1	0	1	0
21:41:50	0	0					1	1	1	0	1	0	1	0
21:41:51	0	0							1	0	1	0	1	0
21:41:52	0	0	1	1					1	0	1	0	1	0
21:41:53	0	0			1	1			1	0	1	0	1	0
21:41:54	0	0					1	1	1	0	1	0	1	0
21:41:55	0	0							1	0	1	0	1	0
21:41:56	0	0	1	1					1	0	1	0	1	0
21:41:57	0	0			1	1			1	0	1	0	1	0
21:41:58	0	0					1	1	1	0	1	0	1	0
21:41:59	0	0							1	0	1	0	1	0

SECTION NO. 4 RECOVERED DATA

GMT	ALTC	ALTF	CAS	HEAD	PITCH	ROLL	N1ENG1	N1ENG2	N1ENG3
21:41: 0	32768	233	296.00	187.38	1.758	0.00	96.68		
21:41: 1		244	295.75	187.03	1.406	0.00		95.25	
21:41: 2		225	295.25	187.03	1.406	-0.70			96.01
21:41: 3		227	295.50	187.03	1.406	-0.70			
21:41: 4	32768	224	295.50	187.03	1.406	0.00	94.70		
21:41: 5		219	295.75	186.68	1.406	0.00		93.29	
21:41: 6		221	295.50	186.68	1.406	-0.35			94.15
21:41: 7		220	295.25	186.68	1.406	-0.35			
21:41: 8	32768	221	295.00	186.68	1.406	-0.70	93.42		
21:41: 9		221	294.75	186.68	1.406	-0.70		92.32	
21:41:10		218	294.75	186.68	1.406	-1.05			93.54
21:41:11		219	294.75	186.68	1.055	-1.05			
21:41:12	32768	221	295.00	186.68	1.055	-0.70	93.32		
21:41:13		223	295.50	186.68	1.055	-0.70		92.26	
21:41:14		231	294.75	186.33	1.055	-0.35			93.45
21:41:15		233	294.00	186.33	1.055	-0.35			
21:41:16	32768	231	293.25	186.33	1.055	-1.05	93.08		
21:41:17		231	293.25	186.68	1.055	-1.05		92.04	
21:41:18		223	293.50	186.68	1.055	-1.05			93.35
21:41:19		221	293.75	186.68	1.055	-0.35			
21:41:20	32768	217	294.75	186.33	1.055	-0.35	93.11		
21:41:21		219	296.00	185.98	1.055	-0.35		91.95	
21:41:22		228	296.75	185.98	1.055	-0.70			93.08
21:41:23		239	297.00	186.33	1.055	-0.70			
21:41:24	32768	238	296.50	186.33	1.055	-0.70	90.85		
21:41:25		228	295.75	186.33	1.055	-0.35		88.47	
21:41:26		216	295.50	186.33	1.406	-0.70			89.23
21:41:27		203	294.75	186.33	1.406	-0.70			
21:41:28	32768	195	293.75	186.33	1.406	-1.05	88.99		
21:41:29		188	293.50	186.33	1.406	-0.70		87.62	
21:41:30		192	293.25	186.33	1.406	-0.70			88.99
21:41:31		194	294.25	186.33	1.406	-0.70			
21:41:32	32768	197	294.00	186.33	1.406	-0.70	89.23		
21:41:33		200	293.75	186.33	1.055	-0.70		87.68	
21:41:34		194	293.75	186.68	1.406	-0.70			89.08
21:41:35		197	293.25	186.68	1.055	0.70			
21:41:36	32768	196	294.75	186.33	1.055	0.70	89.78		
21:41:37		203	295.75	185.63	0.703	-0.35		87.77	
21:41:38		181	298.00	185.98	1.055	-1.05			89.17
21:41:39		215	298.75	185.63	1.055	0.00			
21:41:40	32768	201	297.25	185.63	1.055	-1.05	83.99		
21:41:41		192	296.25	185.63	0.703	-2.46		81.03	
21:41:42		186	293.75	185.98	1.406	-0.35			81.18
21:41:43		184	293.25	185.27	1.406	0.70			
21:41:44	32768	179	292.00	184.57	1.406	0.70	81.27		
21:41:45		182	289.75	184.22	1.406	-0.35		80.63	
21:41:46		176	287.75	184.57	1.055	-1.40			86.46
21:41:47		176	289.00	184.57	1.406	0.35			
21:41:48	32768	213	289.00	183.52	0.703	0.00	94.12		
21:41:49		194	287.75	183.16	1.055	-2.46		93.75	
21:41:50		206	291.25	183.87	1.055	-3.87			97.29
21:41:51		193	291.25	184.57	1.406	-2.11			
21:41:52	32768	182	294.00	184.57	1.055	-1.05	97.69		
21:41:53		182	294.25	184.22	1.055	-0.70		95.70	
21:41:54		180	293.50	184.22	1.055	-2.11			97.11
21:41:55		173	292.25	184.57	1.055	-1.05			
21:41:56	32768	180	293.25	184.57	1.055	-0.70	95.80		
21:41:57		173	294.50	184.57	1.406	0.35		94.03	
21:41:58		182	294.75	184.57	1.055	2.46			94.58
21:41:59		168	296.25	184.22	1.055	-0.70			

SECTION NO. 1 ORIGINAL DATA

GMT	LONG	VERG	VERG	VERG	VERG	LATG	LATG	LATG	LATG
21:41: 0	0.0423	0.9458	0.9824	1.0373	1.0556	0.0174	0.0214	0.0052	0.0174
21:41: 1	0.0397	1.0007	1.0098	1.0007	1.0282	0.0133	-0.0070	0.0011	0.0072
21:41: 2	0.0347	1.0098	0.9915	1.0190	0.9915	0.0052	0.0194	0.0235	0.0235
21:41: 3	0.0403	1.0282	1.0373	1.0465	1.0373	0.0336	0.0296	0.0255	0.0214
21:41: 4	0.0382	1.0098	1.0098	1.0007	1.0007	0.0174	0.0174	0.0113	0.0153
21:41: 5	0.0352	1.0007	1.0098	1.0190	1.0007	0.0113	0.0113	0.0133	0.0133
21:41: 6	0.0336	0.9915	0.9915	1.0007	0.9915	0.0113	0.0133	0.0113	0.0113
21:41: 7	0.0336	0.9915	0.9915	0.9915	0.9915	0.0092	0.0133	0.0092	0.0113
21:41: 8	0.0336	1.0007	0.9915	1.0007	1.0007	0.0133	0.0113	0.0113	0.0113
21:41: 9	0.0336	1.0007	1.0007	1.0007	1.0007	0.0113	0.0092	0.0113	0.0153
21:41:10	0.0341	1.0098	1.0098	1.0098	1.0098	0.0133	0.0133	0.0153	0.0174
21:41:11	0.0331	1.0098	1.0007	1.0007	1.0007	0.0174	0.0133	0.0113	0.0113
21:41:12	0.0321	0.9915	0.9824	0.9824	0.9732	0.0092	0.0153	0.0174	0.0214
21:41:13	0.0316	0.9732	0.9549	0.9549	0.9549	0.0174	0.0214	0.0174	0.0174
21:41:14	0.0316	0.9458	0.9366	0.9366	0.9366	0.0133	0.0133	0.0072	0.0072
21:41:15	0.0306	0.9274	0.9366	0.9458	0.9458	0.0011	0.0031	0.0031	0.0031
21:41:16	0.0326	0.9641	0.9732	0.9915	0.9915	0.0092	0.0133	0.0113	0.0113
21:41:17	0.0341	0.9915	0.9915	1.0007	1.0098	0.0133	0.0153	0.0214	0.0214
21:41:18	0.0347	1.0098	1.0190	1.0190	1.0190	0.0235	0.0194	0.0255	0.0214
21:41:19	0.0341	1.0282	1.0373	1.0465	1.0373	0.0194	0.0194	0.0194	0.0174
21:41:20	0.0326	1.0373	1.0282	1.0190	1.0190	0.0113	0.0113	0.0133	0.0113
21:41:21	0.0301	1.0098	1.0007	0.9915	0.9732	0.0072	0.0092	0.0072	0.0092
21:41:22	0.0260	0.9641	0.9549	0.9641	0.9549	0.0072	0.0092	0.0092	0.0113
21:41:23	0.0235	0.9458	0.9458	0.9366	0.9274	0.0113	0.0113	0.0113	0.0133
21:41:24	0.0219	0.9366	0.9366	0.9458	0.9549	0.0153	0.0153	0.0133	0.0133
21:41:25	0.0230	0.9641	0.9732	0.9824	0.9915	0.0133	0.0133	0.0133	0.0133
21:41:26	0.0255	0.9915	1.0098	1.0190	1.0282	0.0133	0.0113	0.0113	0.0092
21:41:27	0.0275	1.0465	1.0556	1.0648	1.0556	0.0113	0.0133	0.0133	0.0153
21:41:28	0.0286	1.0556	1.0465	1.0556	1.0465	0.0174	0.0174	0.0174	0.0153
21:41:29	0.0260	1.0465	1.0282	1.0282	1.0098	0.0133	0.0133	0.0133	0.0153
21:41:30	0.0255	0.9824	1.0373	1.0373	1.0098	0.0133	0.0113	0.0072	0.0092
21:41:31	0.0199	0.9915	0.9732	0.9641	0.9732	0.0072	0.0174	0.0174	0.0153
21:41:32	0.0250	0.9549	0.9732	0.9915	0.9915	0.0092	0.0072	0.0072	0.0052
21:41:33	0.0219	0.9732	0.9549	0.9458	0.9274	0.0052	0.0113	0.0194	0.0235
21:41:34	0.0219	0.9274	0.9274	0.9183	1.0556	0.0255	0.0214	0.0296	0.0316
21:41:35	0.0357	1.2662	1.1503	1.0007	0.9183	0.0499	0.0581	0.0418	0.0255
21:41:36	0.0270	0.9641	1.1105	1.1503	1.1472	0.0113	-0.0030	0.0031	-0.0193
21:41:37	0.0138	1.0922	0.9824	0.8817	0.8725	0.0031	0.0113	0.0255	0.0255
21:41:38	0.0174	0.9732	1.0373	1.0098	0.8725	0.0357	0.0153	0.0174	0.0255
21:41:39	0.0062	0.7627	0.9274	0.9824	0.9274	-0.0152	0.0275	-0.0091	-0.0254
21:41:40	0.0072	1.0007	1.1105	1.0373	0.8908	-0.0091	-0.0091	0.0214	0.0031
21:41:41	0.0092	0.9041	1.2479	1.1014	0.8634	0.0296	0.0316	0.0804	0.0723
21:41:42	0.0046	0.8634	0.8542	0.7718	0.9000	0.0540	0.0174	0.0052	0.0377
21:41:43	0.0250	1.0922	1.1838	1.0831	0.9824	0.0092	0.0133	0.0235	0.0031
21:41:44	0.0143	0.8817	0.8634	1.0007	1.0373	0.0052	0.0092	-0.0172	-0.0132
21:41:45	0.0214	0.9732	1.0098	1.0648	1.0556	-0.0111	0.0011	-0.0070	0.0214
21:41:46	0.0291	1.0556	0.9366	0.9183	0.8176	0.0296	0.0581	0.0621	0.0642
21:41:47	0.0408	0.9000	1.0190	1.0190	1.0556	0.0906	0.0235	0.0174	-0.0132
21:41:48	0.0387	1.0373	1.0739	0.9732	0.7443	-0.0152	-0.0152	-0.0376	-0.0315
21:41:49	0.0408	0.8542	0.8725	0.9641	1.0465	-0.0274	-0.0498	-0.0355	-0.0254
21:41:50	0.0525	1.1014	1.2021	1.0190	0.8817	0.0133	0.0316	0.0174	0.0153
21:41:51	0.0418	0.8725	0.9366	1.0648	1.1105	0.0174	0.0235	0.0357	0.0397
21:41:52	0.0458	1.0648	1.0648	1.0373	1.0007	0.0397	0.0214	0.0133	-0.0070
21:41:53	0.0428	0.9458	0.9732	1.0739	1.1014	0.0011	-0.0152	-0.0030	-0.0091
21:41:54	0.0448	1.1197	1.0556	1.0465	1.0007	0.0031	0.0255	0.0194	0.0194
21:41:55	0.0418	0.9824	1.0373	1.0007	1.0373	0.0336	0.0113	0.0072	0.0031
21:41:56	0.0316	0.9732	1.0465	0.9366	0.9091	0.0052	-0.0050	-0.0233	0.0133
21:41:57	0.0397	0.9091	0.9091	1.2021	1.1746	0.0296	0.0479	0.0601	0.0296
21:41:58	0.0357	0.9366	1.0190	0.9915	1.0190	-0.0030	0.0377	0.0052	-0.0152
21:41:59	0.0214	1.0922	1.1655	1.1014	0.9091	-0.0274	-0.0091	0.0072	0.0520

SECTION NO. 2 ORIGINAL DATA

GMT	TAT	HSTAB	ELEVLI	ELEVRO	RUDUP	RUDUP	RUDLO	AILLI	AILRO	FLAPR3
21:41: 0		1.402	-0.418	0.450	1.172	0.782	0.652	2.372	-0.199	0.001
21:41: 1	-12.987		0.290	0.300	0.652	0.392	0.164	1.124	-0.199	
21:41: 2		1.402	0.431	0.450	0.001	0.001	-0.259	-0.249	-0.199	0.001
21:41: 3	-12.987		0.148	0.450	0.262	0.522	-0.389	1.124	-0.199	
21:41: 4		1.402	0.431	0.450	0.652	0.652	0.099	1.997	-0.199	0.001
21:41: 5	-12.987		0.007	0.450	0.522	0.262	0.001	0.749	-0.199	
21:41: 6		1.402	0.007	0.450	0.132	0.132	-0.194	0.749	-0.199	0.001
21:41: 7	-12.488		0.148	0.450	0.132	0.132	-0.291	0.874	-0.199	
21:41: 8		1.402	0.148	0.450	0.132	0.132	-0.356	0.624	-0.199	0.001
21:41: 9	-12.987		0.148	0.450	0.132	0.132	-0.356	0.624	-0.199	
21:41:10		1.402	0.148	0.450	0.132	0.132	-0.356	0.375	-0.199	0.001
21:41:11	-12.488		0.148	0.450	0.262	0.262	-0.259	0.500	-0.199	
21:41:12		1.402	0.007	0.450	0.262	0.262	-0.194	0.749	-0.199	0.001
21:41:13	-12.987		-0.277	0.450	0.262	0.392	-0.194	0.624	-0.199	
21:41:14		1.402	-0.135	0.300	0.392	0.392	0.001	1.373	-0.199	0.001
21:41:15	-12.488		0.148	0.300	0.132	0.001	-0.129	0.999	-0.199	
21:41:16		1.402	0.290	0.300	0.001	0.001	-0.519	-0.125	-0.199	0.001
21:41:17	-12.488		0.573	0.300	0.001	0.132	-0.454	0.000	-0.199	
21:41:18		1.402	0.573	0.300	0.262	0.522	-0.226	0.874	-0.199	0.001
21:41:19	-12.488		0.431	0.450	0.522	0.522	0.001	1.373	-0.199	
21:41:20		1.402	0.290	0.600	0.522	0.392	0.001	0.999	-0.199	0.001
21:41:21	-11.988		-0.135	0.450	0.262	0.132	-0.064	0.500	-0.199	
21:41:22		1.402	-0.560	0.450	0.001	0.001	-0.356	0.250	-0.199	0.001
21:41:23	-11.988		-0.277	0.150	0.001	0.001	-0.487	0.500	-0.199	
21:41:24		1.402	0.007	0.300	0.132	0.262	-0.389	0.999	-0.199	0.001
21:41:25	-11.988		0.431	0.300	0.262	0.262	-0.226	0.874	-0.199	
21:41:26		1.402	0.715	0.450	0.262	0.262	-0.194	0.749	-0.199	0.001
21:41:27	-11.988		0.856	0.450	0.132	0.132	-0.259	0.250	-0.199	
21:41:28		1.402	0.998	0.450	0.132	0.262	-0.291	0.250	-0.199	0.001
21:41:29	-11.988		0.431	0.600	0.262	0.262	-0.161	0.874	-0.199	
21:41:30		1.402	0.007	0.600	0.262	0.262	-0.194	0.500	-0.199	0.001
21:41:31	-12.488		0.007	0.450	0.132	0.132	-0.291	0.250	-0.199	
21:41:32		1.402	0.007	0.450	0.132	0.132	-0.291	0.375	-0.199	0.001
21:41:33	-12.488		0.290	0.450	0.001	0.001	-0.356	0.624	-0.199	
21:41:34		1.402	0.007	0.450	0.262	0.392	-0.324	0.375	-0.199	0.001
21:41:35	-12.488		1.989	0.300	0.652	1.433	0.001	3.371	-0.238	
21:41:36		1.402	0.007	0.750	1.303	0.522	0.945	1.498	-0.199	0.001
21:41:37	-12.488		1.281	0.450	0.001	0.001	-0.161	-0.125	-0.199	
21:41:38		1.402	-0.985	0.300	0.262	0.262	-0.389	-0.125	-0.199	0.001
21:41:39	-11.988		-0.418	0.000	0.392	0.132	0.001	2.122	-0.159	
21:41:40		1.402	0.573	0.150	-0.519	-0.389	-0.844	-0.499	-0.159	0.001
21:41:41	-10.989		0.998	0.300	-0.389	0.132	-0.942	-2.746	-0.278	
21:41:42		1.402	-0.135	0.600	1.172	1.433	0.359	1.873	-0.199	0.001
21:41:43	-11.489		0.856	0.150	1.172	0.652	0.880	2.622	-0.199	
21:41:44		1.402	0.431	0.600	0.652	0.262	0.229	1.873	-0.199	0.001
21:41:45	-10.989		0.573	0.450	-0.389	-0.519	-0.552	-0.374	-0.199	
21:41:46		1.402	1.281	0.450	-0.389	0.392	-1.170	-0.249	-0.199	0.001
21:41:47	-11.489		-0.418	0.600	1.693	1.693	0.359	3.371	-0.199	
21:41:48		1.402	1.848	0.300	0.652	-0.259	0.782	0.000	-0.199	0.001
21:41:49	-11.489		-0.985	0.750	-1.040	-1.820	-1.137	-3.495	-0.159	
21:41:50		1.402	1.989	0.000	-1.820	-0.519	-2.536	-2.621	-0.278	0.001
21:41:51	-11.489		-0.418	0.750	0.001	0.262	-0.552	0.749	-0.199	
21:41:52		1.402	1.140	0.150	0.652	0.782	0.001	0.999	-0.199	0.001
21:41:53	-10.989		0.148	0.600	0.522	0.001	0.197	0.250	-0.199	
21:41:54		1.402	0.856	0.600	-0.389	0.001	-0.747	-2.247	-0.199	0.001
21:41:55	-10.989		0.573	0.450	0.262	0.392	-0.389	0.624	-0.199	
21:41:56		1.402	0.290	0.600	0.262	0.001	0.001	0.000	-0.199	0.001
21:41:57	-11.489		-0.843	0.450	0.132	1.042	-0.617	2.122	-0.119	
21:41:58		1.402	1.281	0.300	1.172	1.172	0.782	4.993	-0.199	0.001
21:41:59	-10.490		0.290	0.750	0.001	-0.779	0.132	-3.245	-0.238	

SECTION NO. 3 ORIGINAL DATA

GMT	VHF1	VHF2	TRU1	TRD1	TRU2	TRD2	TRU3	TRD3	S4L1	S4L2	S2L1	S2L2	S4R1	S4R2
21:41: 0	0	0	1	1					1	0	1	0	1	0
21:41: 1	0	0			1	1			1	0	1	0	1	0
21:41: 2	0	0					1	1	1	0	1	0	1	0
21:41: 3	0	0							1	0	1	0	1	0
21:41: 4	0	0	1	1					1	0	1	0	1	0
21:41: 5	0	0			1	1			1	0	1	0	1	0
21:41: 6	0	0					1	1	1	0	1	0	1	0
21:41: 7	0	0							1	0	1	0	1	0
21:41: 8	0	0	1	1					1	0	1	0	1	0
21:41: 9	0	0			1	1			1	0	1	0	1	0
21:41:10	0	0					1	1	1	0	1	0	1	0
21:41:11	0	0							1	0	1	0	1	0
21:41:12	0	0	1	1					1	0	1	0	1	0
21:41:13	0	0			1	1			1	0	1	0	1	0
21:41:14	0	0					1	1	1	0	1	0	1	0
21:41:15	0	0							1	0	1	0	1	0
21:41:16	0	0	1	1					1	0	1	0	1	0
21:41:17	0	0			1	1			1	0	1	0	1	0
21:41:18	0	0					1	1	1	0	1	0	1	0
21:41:19	0	0							1	0	1	0	1	0
21:41:20	0	0	1	1					1	0	1	0	1	0
21:41:21	0	0			1	1			1	0	1	0	1	0
21:41:22	0	0					1	1	1	0	1	0	1	0
21:41:23	0	0							1	0	1	0	1	0
21:41:24	0	0	1	1					1	0	1	0	1	0
21:41:25	0	0			1	1			1	0	1	0	1	0
21:41:26	0	0					1	1	1	0	1	0	1	0
21:41:27	0	0							1	0	1	0	1	0
21:41:28	0	0	1	1					1	0	1	0	1	0
21:41:29	0	0			1	1			1	0	1	0	1	0
21:41:30	0	0					1	1	1	0	1	0	1	0
21:41:31	0	0							1	0	1	0	1	0
21:41:32	0	0	1	1					1	0	1	0	1	0
21:41:33	0	0			1	1			1	0	1	0	1	0
21:41:34	0	0					1	1	1	0	1	0	1	0
21:41:35	0	0							1	0	1	0	1	0
21:41:36	0	0	1	1					1	0	1	0	1	0
21:41:37	0	0			1	1			1	0	1	0	1	0
21:41:38	0	0					1	1	1	0	1	0	1	0
21:41:39	0	0							1	0	1	0	1	0
21:41:40	0	0	1	1					1	0	1	0	1	0
21:41:41	0	0			1	1			1	0	1	0	1	0
21:41:42	0	0					1	1	1	0	1	0	1	0
21:41:43	0	0							1	0	1	0	1	0
21:41:44	0	0	1	1					1	0	1	0	1	0
21:41:45	0	0			1	1			1	0	1	0	1	0
21:41:46	0	0					1	1	1	0	1	0	1	0
21:41:47	0	0							1	0	1	0	1	0
21:41:48	0	0	1	1					1	0	1	0	1	0
21:41:49	0	0			1	1			1	0	1	0	1	0
21:41:50	0	0					1	1	1	0	1	0	1	0
21:41:51	0	0							1	0	1	0	1	0
21:41:52	0	0	1	1					1	0	1	0	1	0
21:41:53	0	0			1	1			1	0	1	0	1	0
21:41:54	0	0					1	1	1	0	1	0	1	0
21:41:55	0	0							1	0	1	0	1	0
21:41:56	0	0	1	1					1	0	1	0	1	0
21:41:57	0	0			1	1			1	0	1	0	1	0
21:41:58	0	0					1	1	1	0	1	0	1	0
21:41:59	0	0							1	0	1	0	1	0

SECTION NO. 4 ORIGINAL DATA

BIBLIOGRAPHY

1. Feltham, R. G., "The Role of Flight Recording in Aircraft Accident Investigation and Accident Prevention," The Aeronautical Journal of the Royal Aeronautical Society, v. 74, p. 573-576, July 1970.

2. Civil Aeronautics Board Bureau of Safety, History and Development of Flight Recorders, by O. E. Patton, January 1966.

3. Allen, B. R. and J. S. Leak, The Potential Role of Flight Recorders in Aircraft Accident Investigation, paper presented at the Aviation Safety Meeting, Toronto, Canada, 31 October - 1 November 1966.

4. Code of Federal Regulations, Title 14, Chapter 1, Paragraph 121.343, Aeronautics and Space, Federal Aviation Administration Flight Recorders, 31 December 1964 as amended.

5. Code of Federal Regulations, Title 14, Chapter 1, Paragraph 37.150, Technical Standard Order TSO-C51a, Aeronautics and Space, Federal Aviation Administration, Aircraft Flight Recorders, 17 November 1974 as amended.

6. National Transportation Safety Board Report NTSB-AAS-75-1, Flight Data Recorder Readout Experience in Aircraft Accident Investigations 1960-1973, Special Study, 14 May 1975.

7. Roberts, C. A., The Digital Flight Data Recorder in Aircraft Accident Investigation, paper presented at the 8th IEEE Computer Society International Conference, San Francisco, California, 26-28 February 1974.

8. Faulkner, D., "The AN/ASH-20(V) Flight Recorder-Locater System," Digest of U. S. Naval Aviation Weapons Systems, May 1970.

9. Roberts, C. A., The Role of the PDP-11 in Aircraft Accident Investigation, paper presented at the Digital Equipment Computer Users Society, San Diego, California, November 1974.

10. Roberts, C. A., The Flight Data Recorder and the NTSB's New Data Reduction Station, paper presented at the 5th International Seminar, Society of Air Safety Investigators, Washington, D. C. 1-3 October 1974.

11. NASA Ames Research Center, Accident Investigation-Analysis of Aircraft Motion from a Limited Set of Radar and Flight Data, by R. C. Wingrove.

12. Aeronautical Radio, Inc., ARINC characteristic 573-6, Mark 2 Aircraft Integrated Data System (AIDS Mark 2), 8 September 1972.

13. Hamilton Standard Division of United Aircraft Corp., HSPC 75E16 Volumes 1 and 2, Aircraft Integrated Data Systems AIDS, Airborne, Equipment, Ground Equipment and Software, 21 April 1975.

14. Tuomela, C. H., The Role of the Flight Simulator in Aviation Safety, report on the meeting of an ad hoc committee of prominent individuals in government and the airline industry held at Naval Postgraduate School, Monterey, California, 18-19 September 1975.

15. U. S. Navy Test Pilot School, Inertially Derived Flying Qualities and Performance Parameters, by W. C. Bowes and R. V. Miller.

16. Hamilton Standard Division of United Technologies, ESP 7517, Crash Data Retrieval Systems for U. S. Military Aircraft, 15 July 1975.

17. Monolithic Systems Corp., Specification Monostore IX/ Planar, CMOS Nonvolatile Semiconductor Memory System, 1975.

18. Nitron Division of McDonnel Douglas, Specification NCM 7010, 1024-Bit Electrically Alterable Non-Volatile MNOS Memory, October 1975.

19. Hoffman, E. J., R. C. Moore, and T. L. McGovern, "Designing a Magnetic Bubble Data Recorder Part 1 -- The Component Level," Computer Design, v. 15, p. 77-85, March 1976.

20. Hoffman, E. J., R. C. Moore and T. L. McGovern, "Designing a Magnetic Bubble Recorder Part 2 -- The System Level," Computer Design, v. 15, p. 99-107, April 1976.

21. McPhillips, A. S., "Inside Microprocessors," Modern Data, v. 8, p. 37-41, January 1975.

22. Ogdin, J. L., "Microprocessors: The Inevitable Technology," Modern Data, v. 8, p. 42-46, January 1975.

23. Twitty, R., "The LSI Microprocessor," Mino-Micro Systems, v. 9, p. 76-78, May 1976.

24. Frankenberg, R. J., "Designer's Guide to: Semiconductor Memories -- Part 2," EDN, v. 20, p. 58-65, 20 August 1975.

25. Allan, R., "Semiconductor Memories," IEEE Spectrum, v. 12, p. 40-45, August 1975.

26. Warner, R. M., Jr., "I-squared L: A Happy Merger," IEEE Spectrum, v. 13, p. 42-47, May 1976.

27. Panigrahi, G., "Charge-Coupled Memories for Computer Systems," Computer, v. 9, p. 33-41, April 1976.

28. Stone, H. S., "The Organization of Electronic Cyclic Memories," Computer, v. 9, p. 45-50, March 1976.

29. Allan, R., "Circuit/System Building Blocks," IEEE Spectrum, v. 12, p. 49-52, January 1975.

30. Allan, R., "Components: Microprocessors Galore," IEEE Spectrum, v. 13, p. 50-54, January 1976.

31. Salzer, J. M., "Bubble Memories - Where Do We Stand," Computer, v. 9, p. 36-41, March 1976.

32. Frankenberg, R. J., "Designer's Guide to: Semiconductor Memories -- Part 1," EDN, v. 20, p. 22-29, 5 August 1975.

33. Frankenberg, R. J., "Designer's Guide to: Semiconductor Memories -- Part 4," EDN, v. 20, p. 62-67, 20 September 1975.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Assoc. Professor U. R. Kodres Computer Science Department Naval Postgraduate School Monterey, California 93940	1
4.	Capt C. H. Tuomela Aviation Safety Programs Naval Postgraduate School Monterey, California 93940	1
5.	LCol G. A. Kerr-Wilson, NDHQ/DAASE 2 National Defence Headquarters Ottawa, Canada K1A OK2	1
6.	Mr. R. C. Wingrove, Code 210-9 NASA Ames Research Center Moffett Field, California 94035	1
7.	Mr. D. W. Althaus Lockheed Aircraft Service Company Box 33 Ontario, California 91761	1
8.	Mr. R. Foley Hamilton Standard Windsor Locks, Connecticut 06096	1
9.	Ms. C. A. Roberts, BAS-11 National Transportation Safety Board Washington, D. C. 20594	1
10.	Capt L. N. Baetz, NDHQ/DAASE 2 National Defence Headquarters Ottawa, Canada K1A OK2	1

thesB1337
Study and design of flight data recordin
3 2768 001 91145 6
DUDLEY KNOX LIBRARY